저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리와의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)를 이해하기 쉽게 요약한 것입니다.

Disclaimer
Abstract

Representing Knowledge in Prototypes and Use Case in Biomedicine

Sungin Lee
Healthcare Management and Informatics
The Graduate School
Seoul National University

Diversity, complexity, and distributed nature of data formats and sources, and the extraordinary amounts of data, have been a great challenge for scientists. Data integration in practice offers at best locally valid integration methods, but with lack of desired flexibility and expandability much demanded for Web data sources. One dominant approach has been ontology-based knowledge representation. Ontology is a science of what exists, and theoretically for a given knowledge domain there can be an unlimited number of alternative ontologies in representing the domain. It is a worldview in which stable and ideally immutable classification of concepts and similarities among objects instantiated from them is constructed. Ontology as knowledge representation has gained significant momentum in science communities; however, due to its structural rigidity and lack of principled development and maintenance mechanisms, it leaves much to be enhanced to be used as a universal tool in accommodating varying knowledge entities, such as blog posts, not just scientifically proven evidence. Prototypes are a well-established research area in cognitive science and psychology for explaining human
cognition. A prototype-based knowledge representation will more closely represent the ways in which humans construct knowledge in a computer-processable manner.

After close examination of current such as ontology, conceptual space, and Frames, this thesis presents a prototype-based knowledge representation model implemented as an ontology as well as prototype knowledge base development approaches with biomedical examples, as well as a simple introduction to prototype learning. The prototype-based knowledge representation model is well positioned to address the drawbacks of the extant knowledge representation models, by offering ready expandability. The simple structure of the new model lowers the barrier for use by knowledge engineers, and fosters a greater freedom of expressions by them. Prototype-based knowledge representation is a new research venue that will open doors to many related research works.

**Keywords:** Knowledge Representation, Prototype, Ontology, Conceptual Space, Metamodel, Frame, Big data, Linked Data, Biomedicine

**Student Number:** 2007-30615
Table of Content

Abstract ...........................................................................................................................................i
Table of Content .........................................................................................................................iii
Figure List .......................................................................................................................................v
1. Introduction .................................................................................................................................1
   1.1. Ontology ...............................................................................................................................1
      1.1.1 Challenges in Ontology .................................................................................................2
         A. Structural Rigidity ............................................................................................................2
         B. Ontology Evolution ..........................................................................................................3
         C. Principled Ontology Development and Maintenance .....................................................4
         D. Inheritance (is-a relation) ...............................................................................................4
         E. Semantic Similarity ..........................................................................................................5
         F. Identity ..............................................................................................................................6
   1.2 Conceptual Spaces ..................................................................................................................8
      1.2.1 Core Model ...................................................................................................................11
         A. Dimension .......................................................................................................................11
         B. Domain ............................................................................................................................12
         C. Property ...........................................................................................................................12
         D. Category Reasoning .........................................................................................................14
   1.3 Frames ..................................................................................................................................15
2. Prototype Theory ..........................................................................................................................16
   2.1 Introduction ............................................................................................................................16
   2.2 Categorical Continuum: Prototypical vs. Classical .............................................................17
   2.3 Prototype and Secondary Effects .........................................................................................20
3. Prototype Model (PM) .........................................................-22
  3.1 Context .................................................................-22
    3.1.1 Contextual Dimensions .......................................-23
      A. Time Dimension ................................................-23
      B. Place Dimension ..............................................-24
      C. Auxiliary Dimensions ........................................-25
  3.2 Prototype Model Constructs ......................................-25
  3.3 Knowledge Building Process .....................................-29
  3.4 Prototype Ontology ................................................-30
    3.4.1 Metaclass ......................................................-30
    3.4.2 Context, Entity, Meta-slot Metaclass ....................-32
  3.5 Knowledge Reuse ..................................................-36
  3.6 Prototype-based Inheritance ....................................-37
4. Prototype Example (SNOMED-CT) ....................................-37
5. Prototype Knowledge Base Development Methods ............-41
  5.1 Hybrid Method (Middle-Out Method) .......................-42
  5.2 Horizontal Method ...............................................-45
  5.3 Prototype Learning ..............................................-47
6. Conclusion ..............................................................-49
References .................................................................-51
국문초록 .....................................................................-56
Figure List

Figure 1. Search Term Combination of Ambiguous Terms --------------------------9
Figure 2. Color Circle --------------------------------------------------------12
Figure 3. Membership Distribution in Prototype-based Category Space --------18
Figure 4. Membership Distribution in Classical Category Space --------------19
Figure 5. A Spandrel at St. Mark’s Cathedral in Venice ----------------------20
Figure 6. Exception (Color) -----------------------------------------------27
Figure 7. Instantiation ------------------------------------------------------28
Figure 8. Exception Strength -----------------------------------------------29
Figure 9. Protégé Metaclass Creating Environment ----------------------------31
Figure 10. Meta-classes and Meta-slots ---------------------------------------32
Figure 11. Metaclass Time ---------------------------------------------------33
Figure 12. Metaclass Entity --------------------------------------------------34
Figure 13. Meta-slots --------------------------------------------------------35
Figure 14. SNOMED-CT Results for the term cancer ---------------------------38
Figure 15. Simple Cancer Prototype Model -----------------------------------39
Figure 16. Multi-category Representation -------------------------------------40
Figure 17. Progressive Prototype KB Development -----------------------------41
Figure 18. The Middle-Out Approach -----------------------------------------42
Figure 19. Cervical Cancer Patient Prototype from UCEC ---------------------44
Figure 20. Ontology as THE Prototypes ---------------------------------------45
Figure 21. Simple Cervical Cancer Prototypes -------------------------------46
Figure 22. Cervical Cancer Patient Prototypes -------------------------------47
Figure 23. Prototype Learning Conceptual View ------------------------------48
1. Introduction

1.1 Ontology

Ontology is a science of what exists in philosophy [1], and theoretically for a given knowledge domain, there can be an unlimited number of alternative ontologies in representing the domain. In other word, an ontological representation of a domain is a worldview in which stable and ideally immutable classification of concepts/classes, and similarity among objects instantiated from them, is constructed. As such, it fosters emphasis on the class defined by a group of shared properties that are true of the class objects. This view is well-understood and conceptually devoid of problems, and provides the theoretical and conceptual foundation for objected-oriented programming and the recent exploration of ontology in knowledge representation.

This Aristotelian worldview, however, is not without challenges. For example to construct a classical representation of a given reality is more of a creative work than an intuitive mechanical maneuvering, and the varied, and, in many cases, subjective ways of determining shared properties for a class make tenuous a possibility of constructing universally-accepted rules upon which to determine these properties. In other words, if one were to take an essential property of a concept as that which provides peculiarity inherent to the concept, it is unavoidable that subjective personal bias arising from individual differences in perceptual systems greatly affects the outcome of resultant rules of classification. This observation entails naturally another, arguably the most serious, hindrance to the classical representation: conceptual gaps that stand between the ontology designers and consumers. There are two areas in which the problem manifests itself:

A. Disparity of content and intent - imagine a class Sodium, as defined by OpenCyc Ontology. The author of the class had “intent” that encompassed (or informed) his
knowledge, reasoning and even intuition for legitimacy of its being constituted as a class and of suitableness of its shared properties. The actual “content” – Sodium class and its accompanying properties- is explicit, but the intent, hidden behind the explicit outcome, is implicit and vague, though the latter may become somewhat explicit to some degree by way of, for example, using documentation on the class, which is unfortunately usually out of reach for class consumers. Hence, the consumer is left to his own discretion and knowledge to probe into the intent;

B. Conceptual tractability – ontology, as appreciated and used in the computing world, suffers from changes or evolution in content, reflecting corresponding alterations in intent, that subsequently demands laborious and sophisticated manipulations of both intent and content of the original representation. Other challenges germane to ontological representation are described below.

1.1.1 Challenges in Ontology

A. Structural Rigidity

Ontology provides a hospital environment for categories of scientifically-verified facts and evidence, with individual concepts being impenetrable divisions of the domain that guarantee mutual exclusivity. This structural rigidity - stemming from its insistence on the absolute satisfaction of necessary and sufficient conditions (properties) - poses great challenges, especially so in disciplines as in genetics that see constantly shifting knowledge. Also conditions that are fundamental in membership determination can be arbitrary and/or simplified – an unavoidable practice since the ontology embodies certain assumptions and contexts of the community it desires to serve [2]. Moreover, an essential character (property) is one which gives some characteristics peculiar to the one being defined, so that it necessarily depends on the perceptual and/or cognitive systems of the designer/developer doing the classification. In other words, an essential property
(or character) deemed to be peculiar to a class is not absolutely free from human
cognitive systems of the person distinguishing the character. In short, the ontology offers
representational stability while sacrificing flexibility. There have been two noteworthy
approaches in which some measure of flexibility has been introduced to ontology
engineering: ontology evolution and principled ontology development.

B. Ontology Evolution

As the term suggests, ontology evolution deals with changes in the ontology.
Changes are brought about to a body of knowledge by discovery of new knowledge
and/or by advancement of techniques that help us delve deeper and in more detail into
hitherto hidden knowledge. Once the changes are substantiated by experts and granted
legitimacy, the ontology is correspondingly modified. Changes in domain knowledge
may be the most common type of change that has to be dealt with in ontology evolution
research. However, other type [3] of change - one incurred from a modified view of the
world or in usage (i.e., change in conceptualization)- is much difficult to deal with.

This type of change typically comes from the goals, tasks and context at hand,
forming the view of the domain distinct from the one that previously informed the
original ontology. It is true that studies on schema evolution [4, 5] have shed much light
to ontology evolution for single ontologies, but ontology engineering that gears towards
development of multiple, cross-organizational ontologies and aims to represent a larger
reality than individual ontologies do, still remains much of an uncharted territory.
Ontology evolution is inevitable because all the potential, intermediate ontologies in the
chain of ontology evolution are simply the past artifacts of our efforts to find the true
reality of the domain. This evolutionary nature of ontology stands in stark contrast to its
structural rigidity, because when as a representation of the reality in which stable and
ideally immutable classification is its innate goal, ontology meets its own challenge
driven from its own innate structural rigidity.
C. Principled Ontology Development and Maintenance

The profusion of ontologies has enabled rich descriptions of knowledge, allowing ever-intricate assembly of knowledge which were otherwise either impossible or required much coordination among experts, to say the least. In order to facilitate streamlined development and subsequent (re)use of ontologies, some measures have to be put into place that covers the whole cycle of ontology design, development, and maintenance. To that end, the OBO (Open Biological and Biomedical Ontologies) Foundry [6] put forth ontology development principles, which have had considerable repercussion in the ways ontologies have been developed, as witnessed in part at BioPortal [7]. The open repository portal now provides many of its biomedical ontologies in the form recommended by the Foundry. As ontology developers have adopted more disciplined practice in ontology development, we may enjoy a higher level of ontology interoperability and maintenance. However, problems may not disappear soon [8] - term redundancy still remains among the OBO ontologies, frustrating the desired orthogonality principle (that is, a term ought to be defined in only one ontology in the Foundry).

D. Inheritance (is-a relation)

The de facto standard to describe the hierarchical structure of concepts in an ontology is to use is-a relation – the closer to the top of the hierarchy the wider the semantic of a class, the closer to the bottom the narrower the semantic. One prominent problem arising from the use of the is_a relation is its universal use. That is, in every step down an is_a hierarchy, the assumption is that all the previous classes prior to the current class enjoy exactly the same identity in both semantic strength and scope. Though an ontology is an excellent example of knowledge captured formally and structurally by way of classes, properties, and relations, they are insufficient in helping understand the
knowledge embedded in the hierarchical structure, particularly so to human users. This
drawback, for example, is evident in the difficulties found in potentially multiple
possibilities of Gene Ontology (GO) annotations [9] to a term, since the inheritance
hierarchy allows legitimacy to all the parent terms in the hierarchical paths to the term as
good candidates, leaving little guidance that would enable us to navigate each step of the
path with graded semantics, so that annotation or matching can be more target oriented.

E. Semantic Similarity

Semantic similarity measurement has been an active research topic in various
departments such as cognitive sciences and psychology [10-13], and in ontology
engineering and Semantic Web [14-19]. And similarity estimation is an essential process
in which humans categorize and infer [20]. In the Semantic Web, semantic similarity is a
foundational research whose outcome paves a firm foundation directly for data
integration and ontology alignment/mapping, and indirectly concept categorization as a
knowledge representation itself. In fact, semantic similarity is essential where uncertain
data is present, for example, by humans as in natural language-based question and
answer systems. Much of the works on ontology-based semantic similarity measures can
be classified in three heads: edge-counting, feature-based, and information content-based
approaches [21]. The edge-counting methods aim to define a function of the path
distance between terms in the hierarchical structure underlying the single ontology.

The feature-based approaches first motivated by the work of Tversky [10] consider
the degree of overlapping between sets of ontological features, suitable for cross-
ontology similarity estimation. The information content approaches use the Information
Theory to define a similarity measure in terms of the degree of informativeness of the
immediate super-concept that subsumes the two concepts being compared.
F. Identity (Owl:SameAs)

The issue of identity is an age-old problem both in philosophy and artificial intelligence. The principle of the identity of indiscernibles by Leibnitz was the first formal definition of identity [22], which shows that if there is no method of distinguishing between what are alleged to be two distinct states of affairs, then there is in fact only one state. Closer to the usual sense of the term identity, it states that there cannot exist two indistinguishable things. More specifically, identity is composed of properties of which a thing is essentially true. Hence, in order for two things to be identical, they must share all the same properties. A question may arise immediately as to this definition: what (essential) properties should be involved in identity statement? A thing may hold true of varying properties depending on contexts in which it finds itself. For example, John is an academic professor in trade, and a father in his family. The domains in which John finds himself dictate some properties to be essential: as a professor, the department he belongs to may be of significance in the domain of profession, whereas as a father, the number of children he is of importance. Also, even in the face of the same properties, one may encounter a problem; for example, take two exact copies of the book “The Odyssey” by Homer. They have exactly the same properties, but you cannot safely make an assertion of identity between them. Even with some notable issues with Leibnitz’s definition of identity, we all are accustomed to asserting identity between two concepts or individuals when they share exactly the same properties. Incidentally, Frege’s notion of identity Rule of Substitution [23] rings close to Leibnitz’s in that the former expresses identity in terms of substitution of name by another, that is, if $a$ and $b$ are identical, $a$ can be substituted for $b$. The issue of identity has recently raised its head again in a significant way in linked data space, in part because the task of asserting identity between datasets has been delegated to individuals in linked data community, linking their own datasets with others. And the main semantic
tool for expressing identity in linked data is owl:sameAs. owl:sameAs “links an individual to an individual”. Such an owl:sameAs statement indicates that two URI references actually refer to the same thing: the individuals (or classes in OWL Full) have the same identity” [24]. This strict notion of identity follows Leibnitz’s principle of identity of indiscernibles, that is, isomorphism of owl:sameAs. However, as with any standard, it would be unreasonable to assume that regular users would strictly use a semantic language construct in the ways it is prescribed in the standard document. Several studies [25-27] have divulged the idiosyncratic uses of identity expressed by owl:sameAs. Halpin and Hayes[25] especially identified four different misuses of identity: “Same Thing As But Different Context”, “Same Thing As But Referentially Opaque”, “Represents”, and “Very Similar To”. Other studies drew and developed upon such variations in order to put forth approaches that introduce nuanced isomorphism by using an identity ontology [26], which uses 8 differentiated properties for representation of non-isomorphic identity (or similarity). Though much clearer and more layered, the ontological approach still leaves the ultimate question of distinction in similarity/identity to individuals by resorting to external vocabularies such as SKOS (Simple Knowledge Organization System) [28], for exactMatch and closeMatch. Though it is true that inference through identity chain in linked data space is rare, it is highly likely that the current practice of using owl:sameAs will not support meaningful semantic integration of linked datasets.
1.2 Conceptual spaces

The current Semantic Web is not semantic enough, especially when it is brought to stand in the light of human cognition, especially in human conceptual representations and reasoning mechanism. It is arguable that the want of a representation that mirrors human cognition and reasoning mechanisms hinders greatly the usefulness of the Semantic Web for individuals.

One key aspect of knowledge representation is arguably the notion of context – i.e., how to represent and process contextual information, which is riddled with semantic similarities, nuances, and sometimes “oxymoronic” combinations of concepts that challenge the semantic power of the current symbolical approach to knowledge representation in the Semantic Web. For instance, take a search for short wait restaurants at Figure 1. This search involves an individual, subjective notion of what it means to be short wait, which may vary individually and culturally depending on context. Ten minutes may become unbearably long for a time-pressed lunch with your co-workers, whereas 30 minutes would be short enough for a relaxing dinner with your friends. Also, what would constitute a restaurant for one may not be a restaurant at all for another, it may be just a simple eating place. The seemingly simple two words ‘short wait’ and ‘restaurant’ are not free from context. This example shows that the Semantic Web would have to incorporate representational muscles that accommodate uncertainty and limited knowledge to become more semantic. Conceptual spaces as a knowledge representation can afford such power that would garnish the Semantic Web with more semantics previously either out of reach, or that involve much laborious tasks.
The notion of conceptual spaces as a framework for knowledge representation was proposed by Peter Gärdenfors [29]. In cognitive sciences, there are three dominating methodologies for representing information: symbolic, associationistic, and conceptual. In the symbolic methodology, cognition is viewed as computation involving symbol manipulation; and the associationism is where associations between information elements carry the main burden of representation. Connectionism is a special kind of associationism that models associations using artificial neuron networks.

Conceptual Spaces is a conceptual approach in which information is represented by geometrical or topological structures rather than by symbols. One of the die-hard dogmas of the Semantic Web is that all semantic contents are reducible to first order logic or to set theory. When compared to how we humans handle concepts, the current class-relation structures of the Semantic Web capture only a small part of what we know about the concepts: typical examples would be the notions of similarity and uncertainty [30, 31] - it is quite cumbersome to express objects based on similarity in a Web ontology language.
In a conceptual space, the semantic similarity of concepts in the space is measured as an inverse function of their distance.

Integration of information from diverse sources has been with us for a long time, prior to the advent of the Semantic Web [32], and the shared ontologies published on the Semantic Web have been used for the interoperation of heterogeneous Web Services by mapping ontologies and supporting ontology translation processes [33]. Tools for schema matching or ontology integration [34] have emerged that cover schema-level, instance-level, element-level, structure-level, and language- and constraint-based matching methods [35]. In general, similarity between entities (classes) housed in different ontologies can be measured by imposing a metric for similarity comparison [36] or using machine learning algorithms that apply probabilities to learn similarities [37].

At present, RDF and OWL are used to describe semantic relation between concepts (and ontologies). The underlying basis of these formal languages is the realist approach, which states that the meanings of concepts are in the real world, that is, there is a (direct) mapping between concepts and the world. This idea is void of the consideration on how we humans diversely understand and use concepts. In contrast, cognitive semantics states that the meanings are mental entities, i.e., cognitive structures in human minds [29].

Conceptual similarity and concept combinations are much more efficiently addressed in conceptual spaces than in the symbolic representation, which by its nature offers no efficient solution to combining concepts that do not share properties. For example, dark snow or open secret combines concepts dark and snow, or open and secret. But the dark or open concept has no intersecting properties with the snow or secret concept. Ad-hoc categories [38, 39] are another example of concept combinations for which ontology languages are not ideal candidates to express, for example, “things to use as defensive weapons when attacked, such as chair, knife, gun, or pencil”, or “things to take from one’s home during a fire”. In case of connectionistic representation (artificial
neuron network), it becomes quite unmanageable for representing concept semantics on the level required for concept manipulation even in a small connectionist model. Well-defined operations to create new concept combinations regardless of shared properties can be applied in conceptual spaces [40]. The following is a general introduction to the essential features of conceptual spaces.

1.2.1 Core Model

There are notions of dimensions and domains, of properties (a special kind of concept) as convex regions in domains, and of concepts defined through properties. In a conceptual space, information is represented as points (individuals) and regions (properties or relations). Semantic structures such as similarity relations can be modelled and measured as distances in the conceptual space.

A. Dimension: A dimension represents various qualities of objects, and dimensions form the framework in which to assign properties to objects and to specify relations among them (examples of dimensions: color, weight, time, etc.) The coordinates of a point in a conceptual space denotes an instance of a dimension, for example, a particular weight. Each dimension is endowed with certain geometrical structures such as topological or ordering structures. For example, if we take the dimension time to be one-dimensional structure, we can easily map it to the line of real numbers. However, identification of dimensions is not an easy matter. In the case of a scientific theory, geometrical structures of its dimensions can be chosen and decided by the scientists proposing the theory. The structures are connected to the measurement methods employed to determine the values on the dimensions. Some dimensions are not so easily attained, and can only be inferred. A dimension is not bound to any specific symbolic representations such as OWL. The primary role of dimensions is to represent various qualities of objects in different domains.
B. Domain: A domain is a set of dimensions (for the sake of convenience, the author skips the difference between integral and separable dimensions). A typical example of a domain is color. The color domain consists of three dimensions: hue, saturation, and brightness. The hue dimension is typically represented by color circle as shown at Figure 2; the saturation dimension ranges from grey to greater intensities; and finally the brightness dimension runs from white to black.

![Color Circle]

Figure 2. Color Circle

Some domains are metric, and some are not, but rather just simple linear ordering.

C. Property: A property is based on a single domain, and a concept is based on multiple domains. This definition is much more convenient and stands in stark contrast to symbolic knowledge representation where the distinction between property and concept is not very clear. More specifically, in a conceptual space, a property is a convex region in some domain, and a concept is represented as a set of convex regions spanning one or more domains. In short, a property is a special case of a concept defined in one domain. For example, blue is represented as a convex region in the color domain. In natural language, adjectives such as tall or large usually refer to a single domain, a property,
whereas nouns such as computer, gene, protein most likely include information about several domains, thus representing concepts.

And it is necessary to note that a concept need not be associated with a closed set of domains; this set may be expanded as one learns about more, if any, aspects of the concept. When multiple domains are involved in a concept, one may want to specify relative weights to the domains. This representation of weights for the domains will depend on the context in which the concept is used. Hence, a concept in a conceptual space contains information about prominence of the domains, and the prominence values change depending on the context. This feature for a concept is not present in the current Web ontologies. Take for example, botulinum toxin (BTX) type A. As a neurotoxic protein produced by the bacterium Clostridium botulinum, the gene that produces the toxin, botA gene, may be the more prominent domain if you are looking at the toxin from a protein sequence and functional information perspective, than if you are using it as a cosmetic treatment option, which would make the median lethal dose (1.3-2.1 ng/kg intravenously, for example) as the more prominent domain.

Also, a concept in a conceptual space is a collection of unrelated properties. There are correlations between the domain regions related to a concept. For example, there is a positive correlation between the sweetness in the taste domain and the sugar level in the nutrition domain [30]. Instances in relation to a concept is a set of points spanning one or more domains that represent individual objects. One key benefit of conceptual spaces as compared to symbolic representation of ontology is that concept hierarchies and classifications are emergent - that is, there is no need to explicitly, a priori, define concept relationships. For example, the convex regions that represent elephant are contained within the convex regions that represent mammal. The convexity of regions naturally permits points in a convex region to have varying degrees of centrality (or similarity), which align conceptual spaces to the prototype theory [41]. One can describe
points in a convex region as more or less central (or prototypical). For example, if a conceptual space is a metric space, one can calculate the center of the region. Note that a space that has a distance function is called a metric space. For example, the Euclidean distance in a two dimensional metric space is $d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$. But this is not true for all convex regions, there can be other shapes such as star-shaped regions.

Just because some members of a category seem to be qualified to be prototypical does not necessarily mean that these members should be the prototypes [42]. Since if a convex region represents a category, the central point simply represents a ‘possible’ object that has the most typical features for the category, but the point (object) does not need to be one of the existing members of the category.

**D. Category Reasoning:** Various category reasoning methods are proposed, such as Voronoi Tessellations (with prototypes) and Region connection Calculus [43], to be used in conceptual spaces. The Voronoi diagram of a set of sites in the plane partitions the plane into regions, called Voronoi regions. The Voronoi region of a site $s$ is the set of points in the plane for which $s$ is the closest site among all the sites. After a Voronoi diagram of a given set of sites has been constructed, the question of assigning a point will become reduced to determining the region containing the point. A Voronoi tessellation will provide divisions of a conceptual space where each object in the space is located with the prototype closest to it. This categorization works fine as long as clear boundaries can be drawn upon a single prototype. This idea of point-based categorization is further expanded by generalizing Voronoi tessellations to include conceptual regions, rather than prototype points [43].
1.3 Frames

The concept of a frame was first proposed by Minsky [44], and frame systems subsequently had gained popularity [45]. A frame is a data structure, or expression, intended to represent a stereotypical situation [46], and it represents an object or concept with slots (or attributes) that can be filled with other data structures, which in turn can be another frame, name or value.

Take an example of a frame representing a typical house [46], with slots for the kitchen, bedrooms, lavatory, address, etc. A particular house (frame instance or an individual) is represented by an instance of the house frame, by filling in the slots with specifications of the parts of the instance of the frame house. For example, the kitchen slot is filled with the frame L-shape-kitchen with a slots lights and cooker, the latter of which again can contain an instance of the modern-cooker frame. Frames allows a certain level of uncertainty in its model in that at any point in time, the model does not require every slot of a frame to be filled, therefore, in the example of the house frame, if the kitchen slot is not known, it can be legitimately omitted.

A frame instance represents an individual, and a slot denotes a relationship that holds between the individual and some other. For example, if a frame instance (f_inst1) of the house frame has its slot kitchen filled with a frame instance (f_inst2), it asserts that there is a kitchen relationship between f_inst1 and f_inst2, or in predicate calculus, “is_kitchen_of (f_inst1, f_inst2)”. In a sense, a frame is a collection of properties and a syntax to express relationships between individuals, as in predicate logic.

Ontologies in the Semantic Web can be created either in description logics-based OWL (OWL-Lite and OWL-DL) or frames such as F-Logic [47]. Both are built on the notion of classes; classes have instances; properties (slots) describe attributes of the classes and relationships between them; restrictions and facets express constraints on the
values of properties and slots [48]. However, differences are notable when it comes to the semantics of these constructs, and how these constructs are used in inference and ontology consistency check, and conversion between them (for example, the Foundational Model of Anatomy (a frame-based ontology) to description logics representation OWL[49]) is still a laborious task.

2. Prototype Theory

2.1 Introduction

The German philosopher Ludwig Wittgenstein developed the notion of family resemblance, which would state that concepts are not defined by stating necessary and sufficient conditions. There may not be even a single feature that is a common for all variants of a term, thus defying the traditional method of ontological definition of the concept (His famous example is the term ‘game’ for which he argues that there is not a single feature that all the games would share in common).

The idea of Wittgenstein’s family resemblance has subsequently developed into prototype theory. The prototype-based view employs alikeness in abstraction, whereas the classical view focuses on categorization based on shared properties. We perceive things in a bottom-up manner – we are better at specification than generalization; we can work better with specific examples from which we can generalize other similar examples. This argument asserts that prototype is closer to our cognitive faculty than class.

Research [41] has shown that there are more “basic” categories than others, and objects are “better” representatives of categories. We humans tend to identity these basic and better, or best, examples first, only after which more general classes can be deduced as our experience or knowledge from the problem domain has increased or strengthened. This is the major difference between the classical knowledge representation and
prototypical. Prototype-based knowledge representation holds only objects, which are worked on much the same as classes are treated in the classical representation. For example, inheritance can be seen as conceptual specification or non-destructive incremental modification of class (class inheritance) or prototype (prototype inheritance). Incremental modification is a process in which new properties are added to refine or defeat certain aspects of parents. Class inheritance occurs obviously between classes, whereas prototype inheritance works directly on objects. One popular way of inheritance for prototypes is cloning in which new properties are dynamically added to create a new prototype.

Prototypical representation is in general pragmatic and contingent upon context, and as such confers into the hands of the designer more latitude and freedom in abstraction. It provides for similarity without strict essentialism, and distinction without sacrificing similarity. Prototypes in essence do not assume any concrete or discrete boundaries in categories, contrary to the classical, ontological approach. And ideally, processes in which category membership is defined can be either fixed for a certain purpose in time and context, or evolves as more domain knowledge and experience are gained. This context-driven flexibility in category membership enables membership inferences to be multifaceted, ranging from stable (i.e., obvious or general) to rare, specialized, and to less obvious and evanescent, effectively supporting varying levels of gradation in category membership. The most prototypical member of a category, so called prototype, holds the most context-relevant characteristics, and candidate members for the category can be aligned or ranked in relation to their similarity to the prototype.

2.2 Categorical Continuum: Prototypical vs. Classical

Categorical continuum of a prototype is informed in general by two gradations: 1) all properties, or features, are weighted in terms of their importance - measurement for individual feature or property; and 2) all members of a category are ranked in terms of
the number of characteristic properties they share - measurement for each potential member. An example of category space, shown at Figures 3 and 4 below, clearly shows the difference of membership distributions for both knowledge representation approaches. Figure 3 shows in resemblance-based categories that there are degrees of similarities among members in a category. It, however, does not necessarily mean that the farther any two members are apart from each other, the less similar they are.

Figure 3. Membership Distribution in Prototype-based Category Space

On the other hand, members of a category in the classical knowledge representation of the Figure 4 reside in a specific location in the category space, assuming only single inheritance allowed.
Members of a category in the prototype can be graded based on their typicality. A good prototype is a good one because of its typical features, that is, a prototype is a collection of typical characteristics of a category. Hence, if we define ‘bird’ having feathers, two wings, two legs, and being able to fly, we can say a chicken is a less-typical bird, since it can’t fly. Some features have higher levels of typicality as well. For example, flying is more typical feature of the bird than its chirping.

Categorization is an adaptive process by which we model a reality. Without it, we fail to understand and make reasonable arguments or predictions about the reality. The classical theory of categorization defines a concept to be a set of necessary and sufficient conditions (features); whereas the prototype theory argues that concepts are prototypes manifesting typical features of objects of a category, negating the necessity and sufficiency conditions of fundamental characteristics put forth by the classical theory. The idea of ‘typicality’ assumes the existence in reality of a phenomenon of typicality when we categorize the reality.
2.3 Prototype and secondary effects

The term ‘exaptation’ defined by Gould [50] in evolutionary morphology describes features that enhances fitness but are not built by natural selection for their current role. Before Gould, the term ‘adaptation’ referred to both 1) features built by natural selection for their present role, and 2) features that enhance fitness regardless of how they arise. Gould however uses the term ‘exaptation’ to designate any feature that is fit for current role, but was not designed for it. It differs from adaptation which designates any feature that promotes fitness and was built by selection for its current role (historical genesis).

Another example of such ‘exaptation’ was put forth as spandrels. Anyone who ever has wandered into a medieval cathedral or mosque may have noticed the beautiful architectural by-products, called spandrels, in erecting a dome on rounded arches of, for example, St. Mark’s Cathedral in Venice, as described in [50] at Figure 5 (http://friendsofdarwin.com/20150420).

Figure 5. A spandrel at St. Mark’s Cathedral in Venice

Spandrels are the results of architectural constraints necessary for building a dome based on rounded arches. They are secondary effects of the original architectural constraints. Spandrels are secondary phenomenon that represents a “fruitful use of
available parts.” In a sense, spandrels represent potentials for an increasing number of
cases in which otherwise previously unforeseen opportunities may unfold in different
times and contexts, in a system designed and developed for other reasons than the new
uses.

An exaptation in the physical world may find its counterpart in knowledge
representation domain. Classical knowledge representation corresponds to adaptation in
evolutionary theory, in that a concept is driven by one driver – commonality of properties
desirably unchangeable and suitable for its current role in the domain. All the instances
of the concept must share the same properties. In contrast, the prototype resembles
exaptation, in that there may be instances that belong to different categories that share the
same properties to a certain extent. For example, chicken as an ingredient for a fried
chicken may have the same properties of a chicken as an animal, except that the former
belongs to food category, and the latter to an animal category. However, they both
belong to the upper category of chicken, which may hold other chicken sub-categories as
‘a chicken as a pet’, ‘a chicken as a coward’, and others. As Gould said, an exaptation is
a feature that was not designed for the role it plays at present. The prototype model under
design aims to provide a means for conceptual spandrels that accommodate such
secondary, undersigned effects by allowing, primarily, loosely defined exceptions in
properties. Exceptions will bridge otherwise seemingly unrelated categories such as ‘a
chicken as a pet’ and ‘chicken as food ingredients’.
3. Prototype Model (PM)

3.1 Context

In antiquity, mathematicians and logicians pursue eternal truths, which does not depend on anything other than the logic they are stated in [51]. Hence, circumstantial facts do not change the truths in this traditional logic. Logics developed by [52] started using situations and contexts. An ordinary person tacitly understands what context is; however, it is not an easy task to elucidate it, let alone put forth a specific and detailed definition of it. It is out of the scope of this thesis to develop in a formal way what constitutes a context. Rather, this thesis has adapted some of the seminal works such as McCarthy [53, 54], Guha [55], and Lenat [56], in order to apply their ideas on context in relation to prototypes. This thesis does not strictly follow the definitions of the works aforementioned; however, the author has endeavored to maintain the main ideas of their work at the same time adding his own qualifications and modifications.

Contexts are treated as entities in the domain, a concept generally introduced by and attributed to John McCarthy [52]. This simply implies that one can make statements about contexts. A context, in general, however, is inherently ‘rich’ in the sense that it cannot be possibly described in full in every sense of the word. An entity is filled with explicit and/or implicit contextual effects that, for one, logic cannot exhaustively capture them. At best, one can as much capture and express contextual information as possible. Context is pivotal in the prototype model, in that it provides the legitimacy of an object being a prototype in the self-same context, though the object may not be considered to be a prototype in another context. In short, an object becomes a prototype only when even incomplete, due to lack of knowledge, contexts dictate it being a prototype.
3.1.1 Contextual Dimensions

There are many contextual dimensions one can think of - time and space may be the two rudimentary but fundamental dimensions of context. The most practical set of context dimensions may be found in Cyc® [55]. This thesis has gained much of its insight form the work on context in Cyc. A brief explanation on common and auxiliary contextual dimensions used in this thesis is given below.

A. Time dimension

Time is a highly sophisticated and at the same time complex concept to represent. What is true at time $t_1$ is not necessarily true at time $t_2$. For example, a stem cell residing in bone marrow at $t_1$ at undifferentiated stage may not be called the same stem cell at $t_2$ when the former is in the process of differentiation into a specialized cell, and while being divided creating other stem cells, the original stem cell may not be considered to hold the same identity at various time spans. A time can be expressed simply, i.e., a point in time such as ‘the year 2106’, or a duration/interval such as ‘during the civil wars of Congo’ which happen to have occurred discontinuously and sporadically over a couple of decades. An interval can be very specific, such as from 3AM to 6PM, or it can be non-specific, circa 1900, which leaves room for certain errors in calculation. A complex time can be a non-absolute time such as ‘after his graduation of the university’. An event can also be specified either in absolute or relative sense in time. For example, when a person purchases a car at a dealership, one can specify the time in an absolute sense as “8:30 AM on Friday, March 1996”, or in a relative sense as “as soon as the door of the building opens”.

When we subscribe an object to a time so that it may hold true in that time, it is normally assumed that the time specified holds true as its entirety for the object. For
example, an object ‘pain’ may a temporal information attached to it such as ‘start time = brought about when the patient fell from a chair’ It is not to say that the pain is felt by the patient exactly at the moment when he or she fell from the chair. Neither does it mean that the patient started to be aware of the pain in 20 seconds or so. It may be useful to have a mechanism in place that helps denote 1) true for the whole duration of time specified, or 2) true for any points, either continuously or discontinuously, of the time specified. Another challenging issue in representing time is attributed to numerous ways we consider a unit in time. For example, a temporal unit in nuclear science may be much smaller than those used in macro-economics. Defining the most appropriate units of time depends on the domain for which a prototype model is being implemented.

This thesis also considers a special aspect of time, namely kairos. The ancient Greeks had two words for time – chronos and kairos. Until now, all the discussions on time is related to the perception of time familiar in our ordinary experience, namely chronos time. Chronos is a sequential and quantified time, whereas kairos is a qualitative, opportune time. For example, the discovery of a new gene could become disruptive in the study of genomics. This time in which such discovery was made would warrant not only a traditional chronos time, but also mark a launch of series of new studies involving the gene. The time dimension in this study thus involves kairos as a sub-dimension of time. The user is free to use this dimension to create a kairos time to denote its significance. The value attached to a kairos time may not be different from a normal chronos time; it is mainly used to denote the importance of the time attached to the object.

B. Place dimension

As with the time dimension, a geographical point is a very important dimension, and can become very complex as well. Geographical locations are ordered, as time can be ordered. The ordering of locations typically entails containment of one or more locations
within another location. A place can be either simple or relative. A simple place is a particular location such as ‘Seoul’. A relative place is a non-absolute location such as ‘at hospital’.

C. Auxiliary dimensions

For some objects, it may be warranted to include certain auxiliary context dimensions. For example, age, generation (ex, baby boomer, post-Korean war, generation X), descendant or provenance (of Korean, of government), work environment (laboratory), and others. Sub-dimensions of these minor auxiliary dimensions can also be created if need be. Another useful but optional dimension may be topic about which a given object is. Topics can be chosen from a variety of international standards such as the Medical Subject Headings (MeSH) at the U.S. National Library of Medicine.

In this thesis, there are three types of context: Core contexts, Optional Contexts, and Domain Contexts. The core and optional contexts are defined to be metaclasses. The domain contexts are regular classes whose superclasses are Context metaclasses. The following shows the Protégé implementation of Context metaclasses.

3.2 Prototype Model (PM) Constructs

The prototype model has purposely a simple knowledge structure, providing three properties for an instance: 1) Intrinsic Property, 2) Extrinsic Property, and 3) Exception. The model is designed to be minimalistic, so that the knowledge building process is to be inherently gradual, building upon potentially a bare minimum amount of knowledge whose prototypical mold is progressively shaped. In fact, it is the core assumption of this model that a prototype is not a concept to be holistically designed a priori, but rather it evolves as the process of continuous knowledge building process persists. This approach
stands in contrast with the ontological knowledge representation, which assumes a priori the existence of potentially immutable and stable structure of knowledge. The conceptualization of the prototype being an evolving (or emerging) entity is a crucial feature that enables multi-context categorization such as ad-hoc categories [38]. Ad-hoc categorization is a clear example of exaptations that are afforded in the prototype model. Before going further, descriptive definitions of model constructs are in order.

Definition 1 (Prototype): A prototype is a better example or instance of a category. More specifically, a prototype in the PM is an instance, or a group of instances, emerged (or derived) by context. That is, an instance is never assumed be a prototype unless it is designated as such, and the determiner of the instance being transformed into a prototype is context. According to this definition, a prototype in one context may not be a prototype in another context.

Definition 2 (Quasi-prototype): A quasi-prototype is an instance whose category membership may be uncertain or undecided at any point in time, and which can’t be a prototype as such. For example, one would be hard-pressed to determine the ‘cellphone’ to be a member of ‘furniture’ category. Most of the instances modelled in the PM can be QPs at one point in knowledge building process (KBP). In fact, all instances are unless specified as a prototype is a quasi-prototype.

Definition 3 (Property): A property of an instance informs one of its distinct characteristics, which are divided into three heads: intrinsic, extrinsic, and exceptional.

Definition 3.1 (Intrinsic Property): An intrinsic property of an instance is a requisite characteristic that partly informs the instance. An entirety of intrinsic properties together informs the minimum, requisite characteristics of the instance.
Definition 3.2 (Extrinsic Property): An extrinsic property is loosely defined in the model. It is considered a “placeholder” for external features that may or may not hold true at a given time or under a specific context.

The nature of an extrinsic property is not clearly defined in this thesis yet, since the nature and values of an extrinsic property cannot be clearly defined in advance. An example of an extrinsic property can be a URI that points to an instance defined outside the domain of the instance being developed. For example, an instance ‘chicken’ is being defined with some intrinsic properties, one of which denotes that chicken is a bird. Imagine that there is another ‘chicken’ instance, defined by another KB developer, that represents the chicken being a type of food, or another being a type of pet, and so on. An extrinsic property of the instance ‘chicken’ as a bird can point to such instances of different ‘chicken’ types. The determination of inclusion of an extrinsic property solely depends on KB developers.

Definition 4 (Exception): An exception of an instance occurs when the value of one of its intrinsic properties of the instance is violated in an intrinsic property of another instance. As shown in the figure below, the LeesCar instance is similar to the JohnsCar instance except for its color being blue. In this case, exception is flagged for color:blue in the LeesCar. And the JohnsCar is a MiddleSizedCar.
As such, an exception always stands in relation to (an) other instance(s). An exception as an act of property violation provides for the very flexibility in the PM by which similar instances to a prototype can be easily represented. It is also assumed that there are varying degrees of significance assigned to exceptions. At this point, no computational model is envisioned for helping users to decide upon a significance weight. It is rather assumed that significance is optional and subjective in nature, not mandatory and objective in its valuation, and thus a user-assigned weight is encouraged for each occurrence of exception.

Definition 7 (Exception Weight): An exception weight of an intrinsic property in an instance can be a user-assigned, relative importance of the exception to the other exceptions in the same instance. However, for the sake of simplicity, it is assumed that every exception in an instance is of equal weight as shown in Figure 8 below, and the more exceptions the less similar.
It is however suggested that the sum of exception weights in an instance should never exceed a total sum of exception weights allowed, for example, 1. Each time a new exception is added to an instance.

### 3.3. Knowledge Building Process

A typical knowledge building process (KBP) of an instance consists of three steps: 1) it is reasonable to assume that the knowledge engineer has access to at least certain stock of knowledge on which he/she endeavors to build the knowledge base. The engineer then starts with the knowledge he possesses, regardless of how insufficient it may seem. He may at this stage specifies some intrinsic and extrinsic properties of an instance. It makes no sense at this point to define exceptions, since exceptions are always defined in relations with other instances. This “start with what you know” approach leaves the nature of KBP progressive and evolutionary, in that the more prominent and collectively agreed-upon intrinsic properties will emerge as the volume of knowledge modelled increases. 2) in the second step, the skeleton QP created in the first step is refined in two
ways. The first step involves making connections of the skeleton with other existing instances. The criteria by which to connect the skeleton to other instances use the idea of “least exceptional instances.” That is, the skeleton is recommended to be linked to an instance that has the least number of exceptions to the former. The number of connections are not limited by the model itself. 3) finally, on the condition that there is a sufficient number of instances present in the knowledge base, the whole knowledge base is refined, so that the overall number of exceptions is always kept to the minimum.

3.4 Prototype Ontology

3.4.1 Metaclass

A meta-class is a template, a class whose instances are classes themselves. Varying conceptualizations, derived from different meta-classes, of a self-same concept are possible by allowing additional information attached to the class. In this thesis, a Frame-based ontology of prototype is developed. One of the major reasons for choosing Frame-based metamodeling is that the most popular ontology language, OWL DL, treats the sets and instances as disjoint, whereas in the Frames, meta-classes are an integral part of its formalism. Also Frames are developed around the notion of prototypes, that is, default values are employed to paint partial knowledge in our perceptions [48]. The Prototype Ontology is developed by using Protégé 3.5 in order to develop metaclasses and some metaslots.
The following essential entities constitute the main body of metaclasses and metaslots as shown in Figure 10.
3.4.2 Context, Entity, Meta-slot Metaclass

There are three different types of context metaclasses: BasicContext, AuxiliaryContext, and PrototypeContext. The BasicContext is a metaclass that represents common contextual information such as Time and Place. The AuxiliaryContext is a metaclass that represents optional contextual information such as Topic (of an object). The PrototypeContext metaclass represents meta-level description of two sub-contextual metaclasses: ExceptionContext and SimilarityContext. The current model is based on exception, instead of similarity for prototype representation.

<BasicContext Metaclass (Time)>

The Time metaclass is a sub-metaclass of the BasicContext metaclass. The Time metaclass has two sub-metaclasses: SimpleTime and ComplexTime. The SimpleTime has 3 slots of importance: 1) PartiallyTrue slot whose value is Boolean and its default value set to false. The default value denotes that the time specified is true in its entire duration; 2) IsKairos slot is a Boolean slot that the user can denote that the time specified has
significance; and 3) TimeValue slot holds the simple time value. The *ComplexTime* has the exactly same definition as the *SimpleTime* except that the TimeValue holds the complex type of time values.

Two sub-metaclasses are defined: *simplePlace* and *RelativePlace*. The *RelativePlace* is a template for all types of relative expression of places such as ‘in the hospital’.

*Figure 11. Metaclass Time*

**<BasicContext Metaclass (Place)>**

Two sub-metaclasses are defined: *simplePlace* and *RelativePlace*. The *RelativePlace* is a template for all types of relative expression of places such as ‘in the hospital’.

**<PrototypeContext Metaclass>**
Both *ExceptionContext* and *SimilarityContext* are defined. The ExceptionContext is the main metaclass that works as a template for representing exceptions in the model. For now, only the allowed values of a property is taken as an exception. There may arise different kinds of exceptions which is not included in the current implementation. For such cases, the metaclass holds room for expressing hitherto unforeseen exceptional cases.

*Entity Metaclass>*

Two sub-metaclasses are defined under the Entity metaclass: Prototype and Quasi-prototype. The Prototype metaclass has a slot with multiple and mandatory cardinality called ByContexts that points to various contextual elements defined in the Context classes, whereas the Quasi-prototype has a slot called MayBelongTo that points to an instance that may be a prototype or an instance to which a property of the instance has an exception. As defined in the model, a prototype is always linked to one or more contexts that promotes it into its being a prototype.
The *Intrinsic* slot defines the most outstanding features of an instance. The *Extrinsic* slot defines an optional property that may or may not be demanded for an instance. It is up to the developer of an instance to include any auxiliary information for this slot. The *ExceptionProperty* is the slot that defines an exception of the instance being developed in relation to either a prototype or another instance.
3.5 Knowledge Reuse

One of the main contributions of prototypes is its bi-directional reuse and sharing: vertical and horizontal sharing. Vertical sharing works by specialization much the same as inheritance of ontologies or vocabularies. A prototype, being specialized by an object in a vertical hierarchy, works as a template of the vocabulary and structure of the new object, mimicking the task ontologies afford. For example, the prototype dentistry (a
typical object being the School of Dentistry at SNU) publishes a collection of prototypes for dental implants related to implant systems and procedures in the prototype dentistry. These implant prototypes can be reused to describe more narrowed-down specific objects. For instance, the prototype “The School of Dentistry at SNU” publishes implant prototypes such as *endosseous dental implant* (implants are placed partially or entirely within the bone) and *intramucosal implant* (implants placed in the soft tissue lining of the oral cavity). Other example implant prototypes include *transendodontic implant* (rod specifically made to be inserted longitudinally either through a root canal or through a root segment), *ramus endosseous implant* (implants placed within the ramus of the mandible), *frame implant* (one-piece implant with a number of mandibular subperiosteal and/or endosseous components), *subperiosteal dental implant* (implant placed beneath the periosteum and overlying the bony cortex), and others. After these prototypes are published, more specific implant objects can be described such as Branemark System MKII as an endosseous dental implant with more specific material dimensions.

Horizontal sharing works by reusing prototypes and changing certain properties or linking to existing prototypes as attribute values. For example, a specific *Branemark System MKII* prototype can be used as a template by describing how other endosseous dental implants made by other implant manufacturers are different from it. By allowing both horizontal and vertical reuse of prototypes, a network of prototypes can be created that can be used over the Web.

### 3.6 Prototype-based Inheritance

The class-based inheritance enables a new class to inherit all properties. However, inheritance in prototype either concatenate or delegate. In concatenation, the new object copy everything from the original prototype and make, if any, necessary changes or exceptions. Delegation enables the new object to maintain a reference to the original
prototype and only keep the changes or exceptions locally. One would argue for either one of the inheritance methods in prototypes. In terms of easy maintenance, I would argue for delegation method, since it would be easier to make changes in the base prototypes without the need to propagate the changes all the down the inheritance hierarchy. Delegation is also more convenient in terms of space, since it does not make physical exact copies of the original prototype.

4. Prototype Example (SNOMED-CT)

SNOMED-CT is a computer processable collection of medical terms related to codes, terms, synonyms and definitions, mostly used in clinical documentation. Arguably the most comprehensive healthcare terminology system, it is maintained by IHTSDO (International Health Terminology Standards Development Organization).

The term cancer returns in the IHTSDO SNOMED-CT Browser 673 terms/concepts that related to the term as shown below at Figure 14. The terms belong to various categories such as disorder, finding, morphologic abnormality and so on.
Figure 14. SNOMED-CT Results for the term cancer

Various kinds of cancer are defined: for example, under the ‘disorder’ category, skin, cecal cancer, lung, renal, liver, mouth, breast, rectal cancers were defined. The term *skin cancer* (malignant neoplasm of skin, SCTID: 372130007) under the ‘disorder’ category has two parent concepts (*Malignant neoplasm of soft tissues* (disorder) and *Neoplasm of skin* (disorder)), and 21 children concepts, further refining skin cancer. This conceptual hierarchy works fine in vertical sharing/reuse, but lack in horizontal reuse.

A very simplified version of cancer prototype model is depicted in Figure 15.
Figure 15. Simple Cancer Prototype Model

Though much details are left out from the figure, the above model shows a much simpler way to represent cancer using prototypes. The above example mimics the cancer concept as a disorder belonging to the category of ‘disorder’ in SNOMED-CT.

There are other categories to which the concept cancer belongs to in SNOMED-CT. For example, *Malignant neoplasm, primary* (SCTID:86049000) is listed under the category of ‘morphological abnormality’. A simple modification of the model above is necessary to represent the morphological information as shown in Figure 16.
The Cancer prototype has an additional property ‘category’. The way in which prototype modeling differs from the classical ontological modelling lies in the fact that the Skin cancer prototype (ID_3) can inherit from the Skin cancer prototype (ID_1), even when the latter is the prototype for all the skin cancer types classified as disorder. The former simply concatenates (or copy) from the ID_1 and change (or raise an exception) for the category value, and all its subsequent children can be generated. This kind of modeling amplifies the power of reuse even from an object that may be deemed to belong to a different kind of grouping (i.e., disorder). Ontological or taxonomical systems are weak in this respect of reusing concept/terms/objects that belong to different domains/categories/groups.

Another point to emphasize is that it is assumed that there does not necessarily exist an is-a relation between the derived prototype Skin cancer prototype (ID_3) from the Skin cancer prototype (ID_1). It simply inherits necessary properties some of which may be modified suitable for the derived prototype. It may even be further assumed that a
patient object created from a patient prototype does not necessarily hold an *is-a* relation. Overall, the *is-a* relation is not enforced in the prototype model.

5. **Prototype Knowledge Base Development Methods**

In this section, an examination of prototype knowledge base development methods is presented. There may be generally two venues considered feasible at this early stage of prototype knowledge base development. Please note that there are currently no available datasets which can make use of features of prototypes such as inheritance. Hence, it is not considered recommendable to blatantly convert RDF datasets to prototypes. It is also assumed that in prototype knowledge base development, you ‘start with what you know, NOT what you must know”. That is, knowledge base development is a progressive task, as depicted in Figure 17. The more essential properties the more exclusions of properties will occur; and the less essential properties the more additions of properties will ensue.

*Start with what you know, NOT what you must know*

![Diagram of progressive prototype KB development](image)

*Figure 17. Progressive Prototype KB Development*
5.1 Hybrid method (middle-out method)

There are two hybrid methods: middle-out method, and ontology-as-prototype method. The middle-out methods aims to extract a set of common and intrinsic features between instance-level data (such as EMR data) and ontology (or other EMR records or description standards), and prototypes are created based on such commons features, as shown in Figure 18.

![Figure 18. The Middle-Out Approach](image)

It, however, should be noted that there may not be any common features shared by instance-level data such as EMR and ontology concepts, because the former describes the clinical observations (such as tumor status and ethnicity) and the latter represents individual clinical concepts in relation to other concepts. The middle-out method is also a
recommended method when there are more than one set of EMR data, in which case those properties common between the EMR data sets are examined to extract the essential properties to create prototypes.

The following example at Figure 19 shows exemplary prototypes based on the uterine cervical and endometrial carcinoma (UCEC) of the Cancer Genome Atlas (TCGA) Pan-Cancer project [57] which gathered data from thousands of patients with tumors, 12 tumor types in total. The clinical patient data published in UCEC have 147 properties such as tumor_status, vital_status, weight, pregnancy count_live_birth, and others.

Many of the values of the properties used by UCEC follow the AJCC (American Joint Committee on Cancer) Cancer Staging Manual (https://cancerstaging.org/references-tools/deskreferences/Documents/AJCC6thEdCancerStagingManualPart2.pdf), especially gynecologic sites (Cervix Uteri), and TNM system for classification of malignant tumors - the TNM stands for the three significant events in the life history of a cancer: local tumor growth (T), spread to regional lymph nodes (N), and metastasis (M). Here the example shows the creation of prototypes for cervical cancer using the AJCC standard and TNM system as sources for essential properties.
In the absence of such common features, it is recommended to take the ontology-as-proto-
type method in which one takes ontological concepts as THE prototypes and attaches other prototypes/instances from instance-level data (EHR), as shown in Figure 20. This method is arguably the harder approach that reuses the current ontological knowledge as the starting point for prototype generation. However, considering the dominance of ontology as knowledge representation, it is a path toward which more research efforts should be directed.
5.2 Horizontal method

This method takes instance-level data as prototypes (or instances), and gradually build prototypes, with or without linking to ontologies. Using the UCEC data above, one could create exemplary prototypes for cervical cancer patient, including ‘tumor_status(values: tumor free, with tumor, unknown)’, ‘tumor_grade (values: G3, G1, G2), ‘histology (values: adenosquamous, cervical squamous cell carcinoma, endocervical type of adenocarcinoma, etc)’, and ‘dx_year (values: 2013, 2006, etc)’. Simple prototypes with 4 essential features with varying values, based on UCEC, would be as shown in Figure 21. The top node contains essential properties with empty values – it is an empty prototype. The three prototypes are provided as explanatory examples. In fact, the brute-force way to create an ‘umbrella’ prototype with all 147 properties is also possible, if not practical.
Institutions which would like to augment these prototypes can do so by adding more properties, for example HPV genotype data, as shown in Figure 22. Once a prototype (or a prototype collection) is decided, all EMR data-driven prototypes can link to them by either delegation or concatenation, with more specific features relevant to them.
5.3 Prototype Learning

A natural course for prototype KB development is (semi-)automatic ways to construct the KB, since it is impractical to expect domain experts to create prototype KBs without computational aids. Prototype Learning is concerned with knowledge acquisition and more specifically with knowledge acquisition from either structured or unstructured texts, that is, prototype population.

Prototype learning can gain much insights from works on ontology learning, which builds on works from natural language processing (NLP), and machine learning. However, the types of data and algorithmic tools for prototype discovering, importing, analyzing, and transforming specific to prototypes would pose challenges different from
those researchers have faced in ontology learning. This thesis simply provides a conceptual view on architecture for prototype learning as shown in Figure 23.

Figure 23. Prototype Learning Conceptual View
6. Conclusion

The complexity, diversity and the distributed nature of data, demand a more efficient and flexible knowledge representation model that accommodate unrelated and (un-)structured data. The classical (ontological) knowledge representation has its strengths, but may not be suitable to deal with the unprecedented amount and types of data that scientists at present have to cope with.

A new knowledge representation method is required – Prototype-based Knowledge Representation. The prototype has been little studied as a knowledge representation model, let alone applied in significant ways for the Web. Prototypes may be treated as an alternative to classical knowledge models such as ontology, especially so since prototypes are more scalable in terms of knowledge sharing and provide more intuitive ways to model knowledge. It, however, would be much more desirable to combine both ontological and prototypical approaches, since the ontological approach seems for now to allow for more solid reasoning capacity, whereas the main thrust of prototypes squarely lies in horizontal and vertical knowledge sharing.

This thesis has put forth 1) a prototype-based knowledge representation model implemented as an ontology; 2) prototype knowledge base development approaches with examples, as well as a simple architectural view on prototype learning.

There are still many research venues related to prototype based knowledge representation. The most significant research directions are 1) to find ways to integrate the prototype knowledge layer on RDF on the current Web stack. The identification of prototypes is relatively easy since all prototypes will be assigned a unique ID (IRI, Internationalized Resource Identifier; or URI, Uniform Resource Identifier), however, once the layer is stacked on the RDF, it remains a challenge to work with prototypes on RDF graphs; 2) how to merge the peculiar strengths of classical knowledge
representation and prototype knowledge representation. That is, the classical has its strength in reasoning and the other in knowledge sharing. It would be challenging tasks to draw clear boundaries between them in ways that would maximize their strengths so that both extant knowledge represented in ontology and emerging prototype knowledge bases complement each other; 3) prototype learning; it may be hard to even picture at this point how prototype learning may develop in the future, but it is a of highly valuable research venue in which a (semi-) automatic way to discover, extract, process prototypes from various resource types such as text, databases, even ontology based knowledge bases is pursued.
References


국문 초록

1. 목 적: 본 연구의 목적은, 현대 데이터 과학자(Data Scientist)들이 직면한, 데이터의 양적 증가, 형식의 다양성, 표현의 복잡성, 데이터 분산 환경 등에 적합한 새로운 지식표현 방법론을 개발/제안하여, 지식의 효과적인 조합과 유연성을 보장하는데 있다. 새로운 지식표현 방법론은 시멘틱 웹(Semantic Web) 기반의 링크드데이터 (Linked Data) 기술과 온톨로지 공학 (Ontology Engineering) 분야, 특히 데이터 중심의 의생명과학 분야에 적합한 지식표현 방법 개발을 목적으로 한다.

2. 방 법: 기존에 존재하는 지식표현 방법들, 즉 온톨로지, 개념적공간(Conceptual Space), 프레임 (Frames)의 구조와 장단점을 분석하여, 새로운 지식표현 방법에 맞는 확장성과 유연성을 강조한 요구사항을 도출하였다. 도출된 요구사항을 반영한 새로운 지식 표현 모델로서, 프로토타입(Prototype) 기반 지식표현 모델을 구축하기 위해 온톨로지 (ontology)를 사용하였고, 개발환경은 메타클래스 (Meta-class)와 메타슬롯 (Meta-slots) 개발을 지원하는 Protege 3.5 온톨로지 개발 환경을 사용하였다. 또한, 프로토타입 지식베이스 구축 방법을 제안하고, 생명의학(Biomedicine)에서의 활용사례를 도출하였다.

3. 결 과: 확장성과 유연성이 강조된 프로토타입 지식 표현 모델을 개발하였으며, 새로운 모델에 기반한 프로토타입 지식베이스 구축 방법을 제안하였고,
생물의학(Biomedicine)에서의 활용사례를 보였다. 본 논문에서 제안한 프로토타입 기반 지식표현 모델은 정형화되고 확장성에 많은 제약을 가진 현재의 지식표현 방식의 한계를 보완하고자 개발된 모델이다. 모델의 구조적 단순성은 지식 구축의 용의성 측면에서 큰 장점이며, 지식 베이스 구축에 있어서 개발자에게 상대적으로 높은 선택적 자유를 보장한다. 프로토타입 지식 표현 모델을 이용한 다양한 지식베이스 구축과 관련 애플리케이션 개발에는 앞으로 더욱 많은 연구가 진행될 것으로 기대된다.

주요어: 지식표현방법, 프로토타입, 온톨로지, 개념적공간, 메타모델, 프레임, 빅데이터, 링크트데이터, 생물의학

학번: 2007-30615