Available Article

Author's final: This draft is prior to submission for publication, and the subsequent edits in the published version. If quoting or citing, please refer to the proper citation of the published version below to check accuracy and pagination.

Cite as: Bergman, M. K. Keeping the Design Open. in *A Knowledge Representation Practionary: Guidelines Based on Charles Sanders Peirce* (ed. Bergman, M. K.) 183–205 (Springer International Publishing, 2018). doi:10.1007/978-3-319-98092-8_9

Official site: https://link.springer.com/book/10.1007/978-3-319-98092-8

Full-text: http://www.mkbergman.com/publications/akrp/chapter-9.pdf

Abstract: The mindset of 'openness' is not a discrete thing, but a concept with separate strands. Open logics and the open-world assumption enable us to add information to existing systems without the need to re-architect the underlying schema. Open-source software has changed the landscape for innovation at low cost. Our knowledge graphs useful to a range of actors must reflect the languages and labels meaningful to those actors.

We should be explicit about the diversity of terms in our vocabularies, using multiple senses to associate related concepts. We use reference concepts to provide fixed points in the information space for linking with external content.

9

KEEPING THE DESIGN OPEN

At the time of my high school years, Alfred Wegener's theory of continental drift was still a question mark for many mainstream scientists. In my college years, a young American biologist, Lynn Margulis, postulated and was ridiculed for the theory of endosymbiosis; that is, that certain cell organelles originated from initially free-living bacteria. In 1980 the Alvarez's hypothesized that the age of dinosaurs was ended by an asteroid strike near the Yucatan at the end of the Cretaceous. In the 1990s we were just starting to get a glimmer the *Helicobacter* bacteria had been the cause of misdiagnosed peptic ulcers for decades. Today, we widely accept all of these then-revolutionary hypotheses as scientific truth.[1]

We now see continental drift as a major explanation for the geographic dispersal of plant and animal families across the globe. Margulis' theory is understood to embrace cell organelles from mitochondria to chloroplasts, informing us that the fundamental unit of all organisms — the cell — is itself an amalgam of archaic symbionts and bacteria-like lifeforms. We now correlate asteroid strikes to historical extinction events through geologic time. Though the native human genome has some 23,000 genes, researchers estimate more than 3 million genes arise from bacterial fellow travelers in our gut and skin 'microbiomes.' We know that our ecosystem of bacteria is involved in nutrition and digestion, contributing perhaps as much as 15% of the energy value we get from food. Besides ulcers, researchers have implicated symbiotic bacteria in heart disease, Type II diabetes, obesity, malnutrition, multiple sclerosis, other auto-immune diseases, asthma, eczema, liver disease, bowel cancer and autism, among others. Within my professional life, major aspects of science, geology, and biology have undergone massive and fundamental shifts in understanding. Concomitant changes have swept through society. Such is the nature of knowledge, with the seeming rapidity of advances steadily increasing.

This chapter begins our *Part III.* All three chapters cover the components of knowledge representation design responsive to such fast-moving changes. In this chapter, we discuss the importance of open design to capture rapid changes in knowledge, indeed to capture the broad trends toward openness across all aspects of human informational and economic activity. These imperatives help inform the structural considerations that go into how to federate and interoperate data from

multiple sources in multiple formats. In the following *Chapter 10,* we discuss our typology design, the basis by which we can adapt our overall design to new domains or expand the knowledge we capture for any given domain. In *Chapter 11*, we explain how these open components naturally also lead to a design founded on knowledge bases and graphs, as the proper structural expressions of this open and connected nature. Think of *Part III*, combined with the three earlier chapters of *Part II*, as describing all of the design and building block inputs needed for a responsive knowledge representation *system,* the topic of *Part IV* that follows.

THE CONTEXT OF OPENNESS

Since ancient times, an exemplar being the Library of Alexandria, humans have used libraries to collate documents and to provide access to knowledge. Repositories and books sometimes threaten authoritarian regimes or close-minded orthodoxies, as does information and knowledge in general. Book burnings and the ransacking of libraries are some of the saddest events of human history.

Fortunately, a notable and profound transition is underway. This transition is not something we can tie to a single year or event. It is also something that is quite complex in that it is a matrix of forces, some causative and some derivative, all of which tend to reinforce one another to perpetuate the trend. The trend that I am referring to is *openness*, and it is a force that is both creative and destructive, and one that in retrospect is also inevitable given the forces and changes underlying it. It is hard to gauge exactly when the blossoming of openness began, but by my lights, the timing corresponds to the emergence of open source software and the Internet. Over the past quarter-century, the written use of the term 'open' has increased more than 40% in frequency in comparison to terms such as 'near' or 'close,' a pretty remarkable change in usage for a more-or-less common term.[2]

An Era of Openness

Though the term of 'openness' is less common than 'open,' its change in written use has been even more spectacular, with its frequency more than doubling (112%) over the past 25 years. The change in growth slope appears to coincide with the mid-1980s,[2] consistent with my thesis of being linked to open source software and the Internet. Because 'openness' is more of a mindset or force — a point of view, if you will — it is not itself a discrete thing, but an idea or concept.[3] In contemplating this world of openness, we can see quite a few separate, yet sometimes related, strands that provide the weave of the 'openness' definition:

- Open source — refers to a computer program in which the source code is available to the general public for use or modification from its original design. Open-source code is typically a collaborative effort where programmers improve upon the source code and share the changes within the community so that other

members can help improve it further;

- Open standards — are standards and protocols, some informal or put forward by individuals, that are fully defined and available for use without royalties or restrictions; stakeholders often suggest and modify these open standards in public collaboration, with adoption subject to some open governance procedures;
- Open content — is a creative work, generally based on text, that others can copy or modify; open access publications are a particular form of open content that provides unrestricted online access to peer-reviewed scholarly research;
- Open data — is the idea that specific data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control; open data is a special form of open content;
- Open knowledge — is what open data becomes when it is useful, usable and used; according to the Open Knowledge Foundation, the key features of openness are availability and access wherein the data must be available as a whole and at no more than a reasonable reproduction cost, preferably by downloading over the Internet;
- Open knowledge bases — are open knowledge packaged in knowledge-base form;
- Open access to communications — is non-discriminatory access to communications networks, allowing new models such as crowdsourcing (obtaining content, services or ideas from a large group of people), citizen science, or crowdfunding (raising funds from a large group of people) to arise;
- Open rights — are an umbrella term to cover the ability to obtain content or data without copyright restrictions and gaining use and access to software or intellectual property via open licenses;
- Open logics — are the use of logical constructs, such as the open world assumption, which enable us to add data and information to existing systems without the need to re-architect the underlying data schema; such logics are essential to knowledge management and the continuous addition of new information;
- Open architectures — are means to access existing software and platforms via such means as open APIs (application programming interfaces), open formats (published specifications for digital data) or open Web services;
- Open government — is a governing doctrine that holds that citizens have the right to access the documents and proceedings of the government to allow for effective public oversight; online access to government data and information is one goal;
- Open education — is an institutional practice or programmatic initiative that broadens access to the learning and training traditionally offered through formal education systems, generally via educational materials, curricula or course

notes at low or no cost without copyright limitations;

- Open design — is the development of physical products, machines, and systems through the use of publicly shared design information, often via online collaboration;
- Open research — makes the methodology and results of research freely available via the Internet, and often invites online collaboration; we refer to it as open science if the research is scientific in nature; and
- Open innovation — is the use and combination of open and public sources of ideas and innovations with those internal to the organization.

In looking at the factors above, we can ask two formative questions. First, is the given item above primarily a *causative* factor for 'openness' or is it a *derivative* due to a more 'open' environment? Second, does the factor have an overall high or low impact on the question of openness. *Figure 9-1* plots these factors and dimensions.

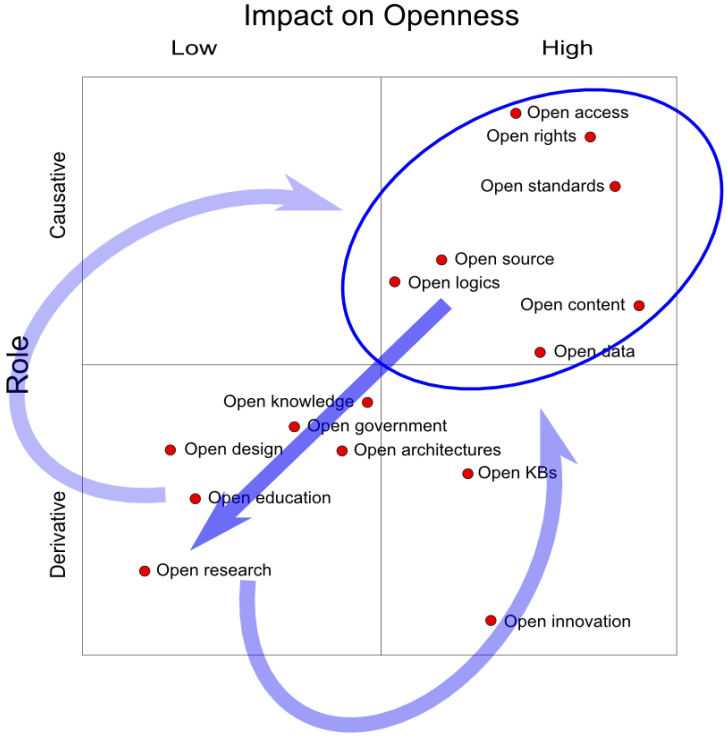

Figure 9-1: Openness Begets More Openness

Early expressions of 'openness' helped cause the conditions that lead to openness in other areas. As those areas also become more open, positive reinforcement is passed back to earlier open factors, all leading to a virtuous circle of increased openness. Though perhaps not strictly 'open,' other various and related factors such as

the democratization of knowledge, broader access to goods and services, more competition, 'long tail' access and phenomenon, and in genuinely open environments, more diversity and more participation, also could be plotted on this matrix.

Once viewed through the lens of 'openness,' it starts to become clear that all of these various 'open' aspects are remaking information technology and human interaction and commerce. The impacts on social norms and power and governance are just as profound. Though many innovations have uniquely shaped the course of human history — from literacy to mobility to communication to electrification or computerization — none appear to have matched the speed of penetration nor the impact of 'openness.' So, what is driving this phenomenon? Where did the concept of 'openness' arise?

The matrix in *Figure 9-1* helps us hypothesize one foundational story. Look at the question of what is causative and what might be its source. One conclusion is the Internet — specifically the Web, as reinforced and enabled by open-source software — is a primary causative factor. Relatively open access to an environment of connectivity guided by standard ways to connect and contribute began to fuel still further connections and contributions. The positive values of access and connectivity via standard means, in turn, reinforced the understood value of 'openness,' leading to still further connections and engagement. More openness is like the dropped sand grain that causes the entire dune to shift. The Web with its open access and standards has become the magnet for open content and data, all working to promote derivative and reinforcing factors in open knowledge, education and government.

The fruits of 'openness' tend to reinforce the causative factors that created 'openness' in the first place. More knowledge and open aspects of collaboration lead to still further content and standards that lead to further open derivatives. In this manner, 'openness' becomes a kind of engine that promotes further openness and innovation. A kind of open logic (premised mainly on the open world assumption, see next section) lies at the heart of this engine. Since new connections and new items are continually arising and fueling the openness engine, we bolt on new understandings to original starting understandings. This accretive model of growth and development is similar to the deposited layers of pearls or the growth of crystals. The structures grow according to the factors governing the network effect, and the nature of the connected growth structures may be represented and modeled as graphs. In general, as might be expected, the greater the degree of structure, the higher its potential contribution to interoperability.

'Openness,' like the dynamism of capitalism, is both creative and destructive.[4] The effects are creative — actually transformative — because of the new means of collaboration that arise based on the new connections between new understandings or facts. 'Open' graphs create entirely new understandings as well as provide a scaffolding for still further insights. The fire created from new understandings pulls in new understandings and contributions, all sucking in still more oxygen to keep the innovation cycle burning. However, the creative fire of openness is also destructive. Proprietary software, excessive software rents, silo-ed and stovepiped information stores, and much else are being consumed and destroyed in the wake of openness.

Older business models — indeed, existing suppliers — are in the path of this open conflagration. Openness is sweeping private and 'closed' solutions into the firestorm. The massive storehouse of legacy kindling appears likely to fuel the openness flames for some time to come.

'Openness' becomes a form of adaptive life, changing the nature, value and dynamics of information and who has access to it. Though much of the old economy is — and, will be — swept away in this destructive fire, new and more fecund growth is replacing it. From the viewpoint of the practitioner on the ground, I have not seen a more fertile innovation environment in information technology. Once the proper conditions for 'openness' were in place, it now seems inevitable that today's open circumstances would unfold. The Internet, with its (generally) open access and standards, was a natural magnet to attract and promote open-source software and content. A hands-off, unregulated environment has allowed the Internet to innovate, grow, and adapt at an unbelievable rate.

Of course, coercive state regimes can control the Internet to varying degrees and can limit innovation. Cybersleuths and hackers may access our information stores and private data, unknown or undetected by us. Any change in the Internet from 'open' to more 'closed' may also act over time to starve the openness fire. Examples of such means to slow openness include imposing Internet regulation, walled gardens like Facebook, limiting access (technically, economically or by fiat), moving away from open standards, or limiting access to content. Any of these steps would starve the innovation fire of oxygen. Access to information wins out over risks, though we do need to self-impose restrictions to guard privacy. Openness reduces the ability of authoritative regimes or close-mindedness to threaten our knowledge.

The forces impelling openness are strong. Still, these observations are no proof for cause-and-effect. The correspondence of 'openness' to the Internet and open source may be a coincidence. However, my sense suggests a more causative role. In all of these regards 'openness' is a woven cord of forces changing the very nature and scope of information available to humanity. 'Openness,' which has heretofore largely lurked in the background as some unseen force, now emerges as a criterion by which to judge the wisdom of various choices. 'Open' appears to contribute more and be better aligned with current forces. Business models based on proprietary methods or closed information appear, at least for today's circumstances, on the losing side of history.

The Open World Assumption

The *open world assumption* (OWA) is a different logic premise for most organizations. Relational database systems, for example, embrace the alternate *closed world assumption* (CWA). OWA is a formal logic assumption that the truth-value of a *statement* is independent of whether or not it is known as true by any single observer or agent. OWA is used in *knowledge representation* to codify the informal notion that in general no single agent or observer has complete knowledge, and therefore cannot make the closed world assumption. The OWA limits the kinds of *inference* and deduc-

tions an agent can make to those that follow from statements known to the agent as true. OWA is useful when we represent knowledge within a system as we discover it, and where we cannot guarantee that we have discovered or will discover complete information. In the OWA, statements about knowledge that are not included in or inferred from the knowledge explicitly recorded in the system may be considered unknown, rather than wrong or false. Semantic technology languages such as *OWL* and *RDF* make the open world assumption. In contrast to the closed-world approach of transaction systems, IT systems based on the logical premise of the open world assumption (OWA) mean:

- Lack of a given assertion does not imply whether it is true or false; it merely is not known;
- A lack of knowledge does not imply falsity;
- Everything is permitted until it is prohibited;
- Schema can be incremental without re-architecting prior schema ('extensible'); and
- Information at various levels of incompleteness can be combined.

Some enterprise circumstances — say a complete enumeration of customers or products or even controlled engineering or design environments — may warrant a closed world approach. CWA is the presumption that what is not currently known as true is false. Engineering an oil drilling platform or launching a rocket, in fact, demands that. A closed-world assumption performs well for transaction operations with easier data validation. The number of negative facts about a given domain is typically larger than positive ones. So, in many bounded applications, the number of negative facts is so large that their explicit representation can become practically impossible. In such cases, it is simpler and shorter to state known 'true' statements than to enumerate all 'false' conditions.

On the other hand, the relational model is a paradigm where the information must be complete, and a single schema must describe it. Traditional databases require we agree on a schema before data can be stored and queried. The relational model assumes that only explicitly represented objects and relationships exist in the domain. It assumes names are unique, and it is how we identify objects in the domain. The result of these assumptions is that relational systems have a *single* (canonical) model where objects and relationships are in a one-to-one correspondence with the data in the database.[5]

It is natural to take a successful approach and try to extend it to other areas. However, beginning with data warehouses in the 1980s, business intelligence (BI) systems in the 1990s, and the general issue of most enterprise information being bound up in documents for decades, the application of the relational model to these areas has been disappointing. CWA and its related assumptions are a poor choice when we attempt to combine information from multiple sources, to deal with uncertainty or incompleteness in the world, or to try to integrate internal, proprietary in-

formation with external data. Irregularity and incompleteness are toxic to relational model design. In the open semantic Web, we can share data that is structured differently via RDF triple statements (*subject – predicate – object*). For example, OWA allows storing information about suppliers without cities and names alongside suppliers with that information. Information can be combined with similar objects or individuals even though they have different or non-overlapping attributes. We now check duplicates based on the logic of the system and not unique name evaluations. Data validation in OWA systems can both become more complicated (via testing against restriction statements) or partially easier (via inference).

It is interesting to note that the theoretical underpinnings of CWA by Reiter[6] arose about the same time (1978) that data federation and knowledge representation (KR) activities also started to come to the fore. CWA and later work on (for example) default reasoning appeared to have informed early work in description logics and its alternative OWA approach. However, the initial path toward KM work based on the relational model also seems to have been set in this timeframe.

We are still reaping the whirlwind from this unfortunate early choice of the relational model and CWA for knowledge representation, knowledge management, and business intelligence purposes. Moreover, while much theoretical and logical discussion exists for alternative OWA and CWA data models, surprisingly few discussions occur for the implications of these models. We may couple the data models behind these approaches (Datalog or non-monotonic logic in the case of CWA; monotonic in the case of OWA; OWA is also firmly grounded in *description logics*) with other assumptions. From a theoretical standpoint, I have found the treatment of Patel-Schneider and Horrocks[5] useful in comparing these approaches. However, the *Description Logics Handbook* and some other varied sources are also helpful.[6] [7]

I think it is fair to assert that the closed world assumption and its prevalent mindset in traditional database systems have hindered the ability of organizations and the vendors that support them to adopt incremental, low-risk means to knowledge systems and management. CWA, in turn, has led to over-engineered schema, too-complicated architectures and massive specification efforts that have led to high deployment costs, blown schedules, and brittleness.

In limited cases, the relational model can embrace the open world assumption, such as the null in SQL. Similarly, semantic Web approaches can be closed world, such as frame languages or Prolog or other special considerations. We can also use relational systems for managing our instance data, while we rely on open world systems for the knowledge graph.

In most real-world circumstances, much we do not know, and we interact in complex and external environments. Knowledge management inherently occupies this space. Ultimately, data interoperability implies a global context. Open world is the proper logic premise for these circumstances. Via the OWA framework, we can readily change and grow our conceptual understanding and coverage of the world, including the incorporation of external *ontologies* and data. Since this can comfortably co-exist with underlying closed-world data, a design based on OWA can readily bridge both worlds. Open world frameworks provide some incredible benefits where

open world conditions apply:

- Domains can be analyzed and inspected incrementally;
- Schema can be incomplete and developed and refined gradually;
- The data and the structures within these open world frameworks can be used and expressed in a piecemeal or incomplete manner;
- We can readily combine data with partial characterizations with other data having complete characterizations;
- Systems built with open world frameworks are flexible and robust; as we gain new information or structure, we can incorporate without negating the information already resident; and
- Open world systems can readily bridge or embrace closed world subsystems.

Open world does not necessarily mean open data, and it does not necessarily mean open source. OWA technologies are neutral to the question of open or public sources. We can apply the techniques equivalently to internal, closed, proprietary data and structures. Moreover, we can use the same technologies as a basis for bringing external information into the organization. Open world is a way to think about the information we have and how we act on it. An open world assumption accepts that we never have all necessary information and lacking that information does not itself lead to any conclusions.

In the past, there have been questions about performance and scalability with open semantic technologies. Progress on these fronts has been rapid, with billion triple systems now common and improvements steady. Fortunately, the incremental approach that we advocate herein dovetails well with these rapid developments. There should be no arguing the benefits of a successful incremental project in a smaller domain, perhaps repeated across multiple domains, in comparison to previous, large, costly initiatives that never produce (even though their underlying technologies are performant). Architecture considerations are also inherent in these OWA designs, which we discuss in *Web-oriented architectures* in *Chapter 12*.

It is perhaps not surprising that one of the fields most aggressive in embracing ontologies and semantic technologies is the life sciences. Biologists and doctors experience daily the explosion in new knowledge and understandings. Knowledge workers in other fields would be well-advised to follow the lead of the life sciences in rethinking their foundations for knowledge representation and management. It is good to remember that if your world is not open, then your understanding of it is closed.

Open Standards

Open standards provide a different kind of openness. The rationale for open standards is not the logic or nature of knowledge, but rather the desire to adopt languages and systems that have the highest likelihood of being shared with others. We employ open standards and best practices in KBpedia to 1) obtain the most accurate

results, and 2) facilitate interoperability with external data and systems.[8] We mostly base our open standards on those from the World Wide Web Consortium (W3C), which established the standards for the original Web and the design of Web pages and Web protocols. Specific W3C standards used by KBpedia include *RDF*, *RDFS*, *OWL 2*, *SKOS*, *SPARQL*, and *SWRL,* introduced in the prior chapter.

Other standards, such as HTML, are also used where appropriate. *De facto* standards may contribute, arising from the effort of individuals or projects. We also may employ open source standard libraries and tools. For KBpedia, these include the ontology IDE, Protégé, the OWL API and the search engine Lucene. In the use of these standards, we apply best practices, many of which we have developed through our client work.* Some of these include the use of *semsets* to capture the multiple labels applied to a given thing; how to construct and manage ontologies (also known as knowledge graphs); ensuring multi-lingual capabilities; and build and management workflows. We discuss these in following sections and chapters. We have written most supporting KBpedia code in Clojure, a modern language based on the original AI language Lisp, in part due to its ability to run on the Java virtual machine. This ability means we may concurrently use any existing Java application with our various KBpedia build, testing, analysis, and management routines.

Open standards, like open source, provide positive feedback across the entire development ecosystem. Developers most often write open source software with open standards and languages. Tooling written in open standards has a broader base of adoption. Developers and knowledge workers prefer to work with open standards because they desire transferable job skills and experience. Like other aspects of the 'openness' phenomenon, open standards are a positive contributor to innovation and still more openness.

INFORMATION MANAGEMENT CONCEPTS

Openness means we also need to accommodate some additional concepts in our design. The first of these considerations relates to how we refer to and name things. Not all of us use the same words for things, and we should be explicit ('open') about this fact in our vocabularies. The second consideration is that we need to provide relatively balanced and equal-weighted concepts in our reference structures. In the case of KBpedia, with its use as a general purpose reference structure, this means we need to capture a set of concepts that capture relatively well the entire knowledge domain. However, the same principles apply to restricted domains and how to define their overall conceptual structure. The third consideration is that, depending on context, we also may use the same term to refer to either an instance or a general class. Again, we should be explicit about these referential differences, with logic and design suitable to them. For lack of a better phrase, I collectively term these three considerations as information management concepts that we need to embrace in our designs.

* See *Chapter 13*.

We intricately associate our vocabularies with how we see and understand the world. We all know the apocryphal claim of how Eskimos have many more words for snow, but the idea likely applies to multiple perspectives in multiple domains. My own first experience is when I was an undergraduate learning plant taxonomy. We had to learn hundreds of strange terms such as *glabrous* or *hirsute* or *pinnate*, all terms of art for how to describe leaves, their shapes, their hairiness, fruits and flowers, and such. What happens, though, when one learns the terminology of a domain is that one's eyes are opened to see and distinguish more. What had previously been for me a field of view composed of multiple shades of green made up of shrubs and trees, began to emerge as distinct species of plants and individual variants that I could discern and identify. As I learned nuanced distinctions, I began to see with greater clarity. In knowledge representation systems, where so much of the knowledge is bound up in text and natural language, training oneself to see the individual leaves and trees from the forest is a critical step to capturing the domain. In part, this attention leads to a richer domain vocabulary.

Things, Not Strings

One of the strongest contributions that semantic technologies make to knowledge-based artificial intelligence (KBAI) is to focus on what things mean, as opposed to how they are labeled. The phrase that captures this focus on underlying meaning is 'things not strings.' The *idea of* something — that is, its *meaning* — is conveyed by how we define that something, the context for how we use the various tokens (terms) for that something, and in the variety of names or labels we apply to that thing. In *Chapter 5*, I provided the examples of *parrots* and the *United States* to illustrate this concept, among other semantic heterogeneities.

We should not view knowledge graphs, properly understood, as being comprised of labels, but of concepts, entities and the relationships between those things. If we construct our knowledge graphs using single labels for individual nodes and relations, we will not be able to capture the nuances of context and varieties of reference. A knowledge graph useful to a range of actors must reflect the languages and labels meaningful to those actors. To distinguish the accurate references of individual terms we need the multiple senses of words to each be associated with its related concepts and then to use the graph relationships for those concepts to help disambiguate the intended meaning of the term based on its context of use.

According to WordNet, a synset (short for *synonym set*) is "defined as a set of one or more synonyms that are interchangeable in some context without changing the truth value of the proposition in which they are embedded."[9] In our view, the concept of a synset is helpful but still does not go far enough. Any name or label that draws attention to a given thing can provide the same referential power as a synonym. If two parties use two different terms to refer to the same thing, we need not go so far as to try to enforce a truth criterion. We can include in this category abbreviations, acronyms, aliases, argot, buzzwords, cognomens, derogatives, diminutives, epithets, hypocorisms, idioms, jargon, lingo, metonyms, misspellings, nicknames,

non-standard terms (see Twitter), pejoratives, pen names, pseudonyms, redirects, slang, sobriquets and stage names as well as, of course, synonyms. Collectively, we call all of the terms that may refer to a given concept or entity a *semset*. In all cases, these terms are mere pointers to the actual something at hand.

In the KBpedia knowledge graph, these terms are defined either as `skos:prefLabel` (the preferred term), `skos:altLabel` (all other semset variants) or `skos:hiddenLabel` (misspellings). *Preferred label* (or *prefLabels* or *title*) is the readable string (name) for each *object* in KBpedia.* We provide labels as a convenience; the actual definition of the object comes from the totality of its description, prefLabel, altLabels, and connections (placement) within the *knowledge graph*. Labels of all kinds are *representations* and reside in Thirdness.

You can inspect for yourself how this concept of semset works in KBpedia. You can go to the standard online KBpedia search page and enter a query, for example, 'mammal.'† By changing between 'Preferred Label' and 'All content' on the dropdown list under 'Search Concepts,' you can get a ten-fold range of results. Naturally, as one would expect, increasing the number of terms something might be known by acts to increase the possible matches within the knowledge graph. Semsets give us a way to narrow or broaden queries, as well as in combination with linked concepts, to disambiguate the context of specific terms. We can apply these same considerations to SPARQL queries or programmatically when working with the KBpedia knowledge graph (or any other graph constructed to KBpedia's standards).

Charles Peirce held strong views about precision in naming things, best expressed by his article on *The Ethics of Terminology*.[10] His beliefs often led him to use obscure or coined terms to avoid poor understanding of common terms. He also proposed a variety of defining terms throughout the life of many of his concepts in his quest for precision. He also understood that terms (*symbols*) could be interpreted in different ways (*interpretants*) by various agents (interpreters). With inquiry, truth-seeking, and the consensus of the community of users, we can reference our desired objects with more precision. That is our ideal. Peirce would concur that many ways refer to the same thing in the real world. The idea of *semset* is expressly designed to capture that insight.

The Idea and Role of Reference Concepts

Interoperability comes down to the nature of things and how we describe those things or quite similar things from different sources. Given the robust nature of semantic heterogeneities in diverse sources and datasets on the Web (or anywhere else, for that matter!), how do we bring similar or related things into alignment? Then, how can we describe the nature or basis of that alignment?

Of course, classifiers since Aristotle and librarians for time immemorial have been

* Other label types may be added to this roster, such as *short-* and *long-labels* that might be the reference for user interface labels where alternatives to *prefLabel* are desired. All labels may also be expressed in any of the standard ISO human languages.

† See http://kbpedia.com/knowledge-graph/search/?query=mammal&index=rcs.

putting forward various classification schemes, controlled vocabularies and subject headings. When one wants to find related books, it is convenient to go to a central location where we may find books about the same or similar topics. If we can categorize the book in more than one way — as all are — then something like a card catalog is helpful to find additional cross-references. Every domain of human endeavor makes similar attempts to categorize things. On the Web we have none of the limitations of physical books and physical libraries; locations are virtual, and copies can be replicated or split apart endlessly because of the virtually zero cost of another electron. However, we still need to find things, and we still want to gather related things together. As stated by Elaine Svenonius, "Organizing information if it means nothing else means bringing all the same information together."[11] This sentiment and need remain unchanged whether we are talking about books, Web documents, chemical elements or our information stores.

Like words or terms in human language that help us communicate about things, how we organize things needs to have an understood and definite meaning, hopefully, bounded with some degree of precision, that enables us to have some confidence we are communicating about the same something with one another. However, when applied to the Web and machine communications, we need further precision in characterizations and definitions.

The notion of a Web basis organizing things is both easier and harder than traditional approaches to classification. It is easier because everything is digital: we can apply multiple classification schemas and can change them at will. We are not locked into legacy structures like huge subject areas reserved for arcane or now historically less relevant topics, such as the Boer Wars or phrenology (though we still accommodate access). We need not move physical books around on shelves to accommodate new or expanded classification schemes. We can add new branches to our classification of, say, nanotechnology as rapidly as the science advances. The notion is harder because we can no longer rely on the understanding of human language as a basis for naming and classifying things. Actually, of course, language has always been ambiguous, but it can be manifestly more so when put through the grinder of machine processing and understanding. Machine processing of related information adds the new hurdles of no longer being able to rely on text labels ('names') alone as the identifier of things and requires we be more explicit about our concept relationships and connections. Fortunately, here, too, much has been done in helping to organize human language through such lexical frameworks as WordNet and similar. We have learned much while grappling with these questions of how to organize and describe information to aid interoperability in an Internet context.

One formalized approach has been put forward by the FRSAD (Functional Requirements for Subject Authority Data) working group,[12] a community of librarians and information specialists, dealing with subject authority data. Subject authority data is the type of classificatory information that deals with the subjects of various works, such as their concepts, objects, events, or places. As the group stated, the scope of this effort pertains to the 'aboutness' of various conceptual works. The framework for this effort, as with the broader FRBR effort, are new standards and ap-

proaches appropriate to classifying electronic bibliographic records. The FRSAD approach distinguishes the *idea of something* (which it calls a *thema*, or entity used as the subject of a work) from the name or label of something (which it calls *nomen*). For many in the logic community, steeped in the Peirce triad of *sign–object–interpretant*,[13] this distinction seems rather obvious and straightforward. However, in library science, labels have been used interchangeably as identifiers, and making this distinction clean is a real contribution. The FRSAD effort does not discuss how the thema is found or organized.

The notion that we use for a *reference concept* contains elements of this approach. A *reference concept* (RC) is *the idea of something* or a *thema* in the FRSAD sense. However, as we use it, an RC is also a reference linking point for external sources or expanded vocabularies. If properly constructed and used, a reference concept becomes a fixed point in an information space. Think of an RC as a fixed starting point for navigating, relating, or mapping content. It is a guiding star in a constellation of information, or, to use a different analogy, a defined, fixed survey marker as used by surveyors to measure new mapping points. As one or more external sources link to these fixed points, it is then possible to gather similar content together and to begin to organize the information space, in the sense of Svenonius. Further, if the RC is itself part of a coherent structure, then additional value can be derived from these assignments, such as inference, consistency testing, and alignments. If the right factors are present, it should be possible to relate and interoperate multiple datasets and knowledge representations.

We have six requirements for a reference concept, some provided by RDF or OWL:

1. *Persistent IRI* – by definition, a Web-based reference concept should adhere to linked data principles and should have an IRI as its address and identifier. Also, by definition as a 'reference,' the vocabulary or ontology in which the concept is a member should be given a permanent and persistent address. Steps should be taken to ensure 24×7 access to the RC's IRIs since external sources will be depending on them. As a general rule, the concepts should also be stated as single nouns and use CamelCase notation (that is, class names should start with a capital letter and not contain any spaces, such as `MyNewConcept`);
2. *Preferred label* – provide a preferred label annotation property that is used for human readable purposes and in user interfaces. For this purpose, a construct such as the SKOS property of `skos:prefLabel` works well. Note, this label is *not* the basis for deciding and making linkages, but it is essential for mouseovers, tooltips, interface labels, and other human use factors;
3. *Definition* – give all RCs a reasonable definition, since that and linkages are what gives an ontology its semantics. Remember not to confuse the label for a concept with its meaning. For this purpose, a property such as `skos:definition` works well, though others such as `rdfs:comment` or `dc:description` are also commonly used. The definition, plus linkages to other concepts, are the two most critical sources for the concept's meaning. Adequate text and

content also aid semantic alignment or matching tasks;

4. *Semset* – include explicit consideration for the idea of a '*semset*' as described above, which means a series of alternate labels and terms to describe the concept;
5. *Language independence* – keep the identifier separate from its labels, and qualify entries for definition, preferred label, and alternative labels with language tags. Though an additional step (for example, assigning the RDF `xml:lang="en"` tag for English), adhering to this practice gives language independence to reference concepts. Sources such as Wikipedia or Wikidata, with their richness of concepts and multiple language versions, can then be a basis for creating alternative language versions; and
6. *Part of a coherent structure* – test for consistency and coherence when modifying the knowledge structure. A cohesive structure provides the benefits of reliable inferencing, discovery, navigation, and analysis. Adequately constructed RDFS and SKOS data models and OWL ontologies can deliver these benefits.

To this basic set of reference concepts, it is also necessary to add the mapping predicates that relate the RCs to external sources. The mapping predicates have their own set of design guidelines:

1. Provide the same *completeness of specification* as RCs;
2. Capture a spectrum of *mapping alignments* from `exact` or `sameAs` to approximate to represent the real correspondence between items; and
3. *Range and domain* – use domains and ranges, as provided for by RDFS, to assist testing, disambiguation, and external concept alignments. Domains apply to the subject (the left-hand side of a triple); ranges to the object (the right-hand side of the triple).

In part, many current vocabularies meet these guidelines to some extent. However, few vocabularies provide complete coverage, and across a broad swath of domain needs, gaps remain. This unfortunate observation applies to upper-level ontologies, reference vocabularies, and domain ontologies alike.

KBpedia is a knowledge graph of approximately 55,000 reference concepts designed according to these design guidelines. We organize its reference concepts into about 80 modular and distinct (mostly disjoint) *typologies*, which I discuss in some detail in the next *Chapter 10*. The RCs that represent the top-level nodes of these typologies we also term *SuperTypes* (also Super Types), which are collections of (mostly) similar reference concepts. We design most of the SuperType disjoint from the other SuperType classes. Each typology we use in KBpedia thus has its corresponding top-level SuperType node.* SuperTypes, and the typologies they represent, thus provide

* In KBpedia, disjoint SuperTypes are termed 'core', other SuperTypes used mostly for organizational purposes are termed 'extended'. KBpedia has a total of about 80 SuperTypes, with 30 or so deemed as 'core'. See further *Appendix B.*

a higher-level of clustering and organization of the reference concepts. The KBpedia Knowledge Ontology (KKO) only contains the highest-level RCs and SuperTypes. This design enables a higher level view of KBpedia with only a couple hundred RCs and makes clear these SuperType typology tie-in points.

For specific domain purposes, you may use KBpedia or portions thereof as the initial grounding structure. You may expand it into new domain areas following similar design considerations. Potentially, you may turn nearly any of the existing 55,000 RCs in KBpedia into a SuperType, providing a new tie-in point to the new RCs reflecting the expanded domain.

Punning for Instances and Classes

In ontologies, we may want to treat our concepts as both classes and instances of a class. *Punning*, in computer science, refers to a programming technique that subverts or circumvents the type system of a programming language, by allowing us to treat a value of a particular type as a value of a different type. When used for ontologies, it means to treat a thing as both a class and an instance, with the use depending on context. To better understand why we should pun, let's look at a couple of examples, both of which combine organizing categories of things and then describing or characterizing those things. This dual need is common to most domains.

For the first example, let's take a categorization of apes as a kind of mammal, which is then a kind of animal. In these cases, ape is a class (general), which relates to other classes, and apes may also have members, be they particular kinds of apes or individual apes. At the same time, we want to assert some characteristics of apes, such as being hairy, two legs and two arms, no tails, capable of walking bipedally, with grasping hands, and with some being endangered species. These characteristics apply to the notion of apes as an instance. As another example, we may have the category of trucks, which we further split into truck types, brands of trucks, type of engine, and so forth. Again, we may want to characterize that a truck is designed primarily for the transport of cargo (as opposed to automobiles for people transport, both of which are vehicles), or that trucks may have different drivers license requirements or different license fees than autos. These descriptive properties refer to trucks as an instance. These mixed cases combine both the organization of concepts and their relations and set members with the description and characterization of these concepts as things unto themselves. This dual treatment is a natural and common way to refer to things for most any domain of interest.

Prior practice has been to represent these mixed uses in RDFS or OWL Full, which makes them easy to write and create since most 'anything goes' (a loose way of saying that the structures are not decidable).[14] OWL 2 has been designed to fix this by adding punning, which is to evaluate the object as either a class or individual based on contextual use; the IRI is shared, but we may view its referent as either a class or instance based on context. Thus, we allow the use of objects both as concepts (classes) and individuals (instances), the knowledge graph is decidable, and we may use standard OWL 2 reasoners against them.[15]

We can diagrammatically show this instance-class dual-use of punning as follows:

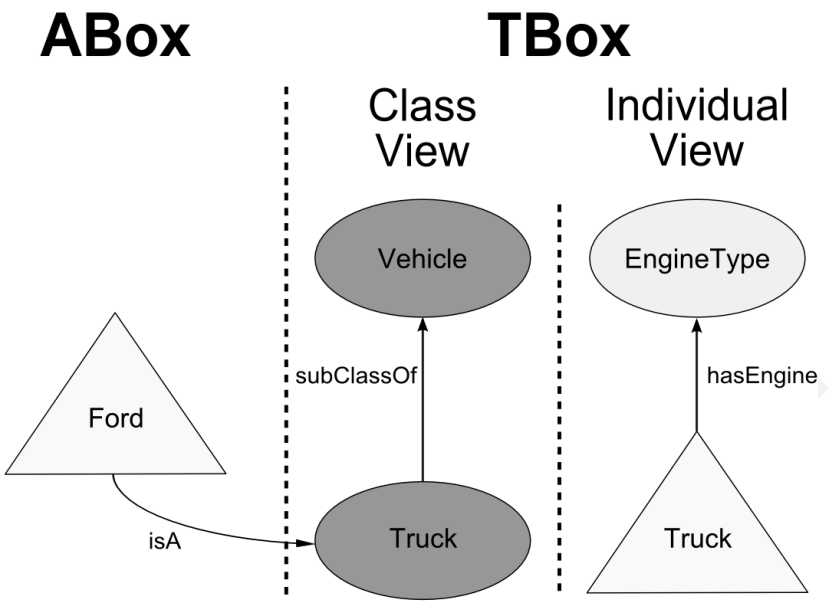

Figure 9-2: Example of Punning 'Truck'

TAMING A BESTIARY OF DATA STRUCTS

The real world is one of heterogeneous datasets, multiple schema, and differing viewpoints. Even within single organizations — and those which formerly expressed little need or interest to interoperate with the broader world — data integration and interoperability have been a real challenge, as we discussed in *Chapter 4*. We should view simple instance record assertions and representations — the essence of data exchange — separately from schema representations. Data values and attributes pose similar problems to that of concepts when trying to get datasets to interoperate. Like dictionaries for human languages, or stars and constellations for navigators, or agreed standards in measurement, or the Greenwich meridian for timekeepers, we need fixed references to orient and 'ground' each new dataset over which we attempt to integrate. For data values, symbol grounding means that when we refer to an object or a number — say, the number 4.1 — we are also referring to the same metric. 4.1 inches is not the same as 4.1 centimeters or 4.1 on the Richter scale, and object names for set member types also have the same challenges of ambiguous semantics as do all other things referred to by language. Without such fixities of reference, everything floats concerning other things, the cursed 'rubber ruler' phenomenon. In *Chapter 5* we discussed the wide variety of formats and data *structs* in the wild, noting specific design approaches might be embraced to help. We address how to tame this diversity in this section, at the same time putting our data onto a common framework.

Rationale for a Canonical Model

In the context of data interoperability, a critical premise is that a single, canonical data model is highly desirable. Why? Because of $2N$ v 2^N. That is, a single reference ('canon') structure means that fewer tool variants and converters need be developed to talk to the myriad of data formats in the wild. With a canonical data model, talking to external sources and formats (N) requires only converters to and from the canonical form ($2N$). Without a canonical model, the exponential explosion of needed format converters becomes 2^N, meaning that every format needs to have a converter to and from all of the other formats.[16] For example, without a canonical data model, ten different formats would require 1024 converters; with a canonical format, 20 (assuming bi-directional converters).

A canonical data model merely represents the agreed-upon internal representation. It need not affect data transfer formats. Indeed, in many cases, we may employ different internal data models from what we use for data exchange. Many data systems, in fact, have two or three favored flavors of data exchange such as XML, JSON or the like. In most enterprises and organizations, the relational data model with its supporting RDBMs is the canonical one. In some notable Web enterprises — say, Google — the exact details of their internal canonical data models are hidden from view, with APIs and data exchange standards being the only portions visible to outside consumers. Generally speaking, a canonical, internal data standard should meet a few criteria:

- Be expressive enough to capture the structure and semantics of any contributing dataset;
- Have a schema itself which is extensible;
- Be performant;
- Have a model to which it is relatively easy to develop converters for different formats and models;
- Have published and proven standards; and
- Have sufficient acceptance to have many existing tools and documentation.

Other desired characteristics might be free or open source tools, suitable for much analytic work, efficient in storage, and easy for users to read and maintain.

The RDF Canonical Data Model

Many wild data forms are patently inadequate for modeling and interoperability purposes. That is why many of these simpler forms might be called 'naïve': they achieve their immediate objective of simple relationships and communication, but require understood or explicit context to meaningfully (semantically) relate to other forms or data. However, besides naïve forms, two common formats with many variants also should be explicitly considered: the entity-attribute-value (EAV) model and

RDBM systems.

EAV is a data model to describe entities where the number of *attributes* (properties, parameters) that can be used to describe them is potentially vast, but the number that may apply to a given entity is relatively modest. In the EAV data model, each attribute-value pair is a fact describing an entity. EAV systems trade off simplicity in the physical and logical structure of the data for complexity in their metadata, which, among other things, plays the role that database constraints and referential integrity do in standard database designs.

On the other hand, RDBMSs use the relational model and store their data in a tabular form, with rows corresponding to the individual data records and the columns representing the properties or attributes. RDF can be modeled relationally as a single table with three columns corresponding to the *subject–predicate–object* triple. Conversely, a relational table can be modeled in RDF with the *subject* IRI derived from the primary key or a blank node; the *predicate* from the column identifier; and the *object* from the cell value. Because of these affinities, it is also possible to store RDF data models in existing relational databases. (In fact, many RDF 'triple stores' are RDBM systems with a tweak, sometimes as 'quad stores' where the fourth tuple is the *graph*.) Moreover, these affinities also mean that RDF stored in this manner can also take advantage of the historical experience gained from RDBMS performance and SQL query optimizations.

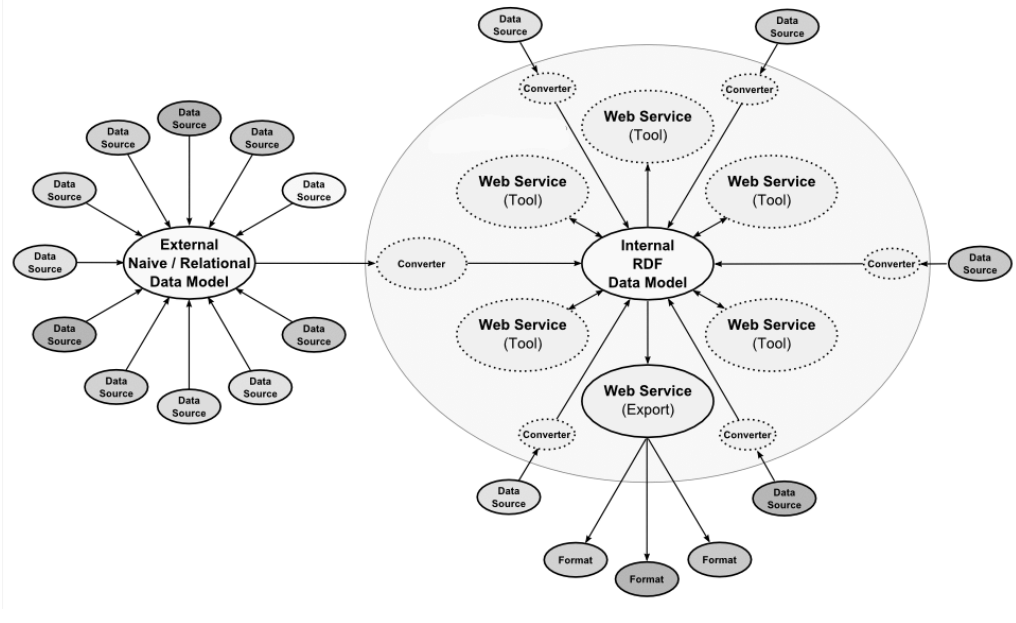

Figure 9-3: Interoperability and RDF as the Canonical Data Model

RDF (Resource Description Framework) might be called a superset of these two forms and is exquisitely suited to accommodate them. In fact, because of its flexible data structure ranging from implied EAV through both of these forms and including

schema as well, RDF is a kind of 'universal solvent' that can readily model most any known data form.[17] When we match this flexible format representation with the ability to handle semantic differences through ontologies at the OWL 2 level, it is clear why RDF provides a competent data model around which to build an interoperable framework. Moreover, because we give all of the information unique Web identifiers (IRIs), and the whole system resides on the Web accessible via the HTTP protocol, our information may reside anywhere the Internet connects. These are the reasons why we have chosen RDF as our canonical data model.

Figure 9-3 on the prior page shows how we approach data interoperability. The input bubbles at the left of the diagram represent the different data formats that the system must address. We process each through a converter to RDF, which is the internal form used on the right. We map schema information associated with each left-side external source to this internal RDF (and OWL 2) data model in advance, before ingesting content.* Note that the converter responsible for the external content ingest may (should!) be a callable Web service so that external sources may call for or schedule ingests according to security standards.

Converters (also known as translators or RDFizers)[18] are an essential bridge to this external world, which we should design for re-use and extensibility. While some may be one-off converters (sometimes off-the-shelf RDFizers), and often devoted to large volume external data sources, it is also helpful to emphasize one or more 'standard' naïve external formats. A 'standard' external format allows for a more sophisticated converter and enables specific tools more easily justified around the standard naïve format. In today's environment, that 'standard' may be JSON or a derivative; or new standards as they arise. Other common 'naïve' formats could be SQL from relational databases or other formats familiar to the community at hand.

In many ways, because we emphasize the *ABox* and instance records and assertions in data exchange, the actual format and serialization is pretty much immaterial. Emphasizing one or a few naïve external formats is the cost-effective approach to tools and services. Even though the format(s) chosen for this external standard may lack the expressiveness of RDF (because the burden is principally related to data exchange), we can readily optimize this layer for the deployment at hand.

Other Benefits from a Canonical Model

As we can see in *Figure 9-3*, converters may themselves be *bona fide* Web services. Besides import converters, it is also useful to have export services for the more broadly used naïve external formats. Exporters allow us to share data and schema with external applications, up to the full expressiveness of RDFS, SKOS or OWL 2. We may devote other services to data cleanup or attribute (property) or object reconciliation (including disambiguation). In this manner, we can improve the authority and trustworthiness of installations, while promoting favored external data standards. Another common service is to give naïve data unique IRI identifiers and to make it

* Depending on the nature of the new external content, it may also be necessary to update the *knowledge graph* at this point.

Web-accessible, thus turning it into linked data.

Such generic services are possible because the canonical RDF model is the 'highest common denominator' for the system. Because RDF is the consistent basis for tools and services, once a converter is available, and we have mapped the external information schema to the internal structure, we can re-use all existing tools and services. Moreover, we are now ready to share this system and its datasets with other instances, within the organization and beyond.

Chapter Notes

1. Some material in this chapter was drawn from the author's prior articles at the *AI3:::Adaptive Information* blog: "Open Source Business Models" (Aug 2005); "Open Source and the 'Business Ecosystem'" (Aug 2005); "Climbing the Data Federation Pyramid" (May 2006); "The Open World Assumption: Elephant in the Room" (Dec 2009); "Listening to the Enterprise: Total Open Solutions, Part 1" (May 2010); "Metamodeling in Domain Ontologies" (Sep 2010); "What is a Reference Concept?" (Dec 2010); "Declining IT Innovation in the Enterprise" (Jan 2011); "We Are an Open World" (Sep 2012); "The Era of Openness" (Jan 2015); "The Importance of Semsets in Knowledge Graph Design" (Mar 2017).
2. The data is from Google book trends data based on this query (https://books.google.com/ngrams/graph?content=open%2Cclose%2Cnear%2Copenness&case_insensitive=on&year_start=1980&year_end=2008); the years 2009 to 2014 were projected based on prior actuals to 1980; percentage term occurrences were converted to term frequencies by 1/n.
3. Stanley, K. O., Lehman, J., and Soros, L., "Open-Endedness: The Last Grand Challenge You've Never Heard Of," *O'Reilly Media* Available: https://www.oreilly.com/ideas/open-endedness-the-last-grand-challenge-youve-never-heard-of.
4. "Creative destruction" is a term from the economist Joseph Schumpeter that describes the process of industrial change from within whereby old processes are incessantly destroyed and replaced by new ones, leading to a constant change of economic firms that are winners and losers.
5. Patel-Schneider, P. F., and Horrocks, I., "Position Paper: A Comparison of Two Modelling Paradigms in the Semantic Web," *Proceedings of the 15th international conference on World Wide Web*, ACM, 2006, pp. 12–12.
6. Baader, F., Calvanese, D., McGuiness, D., Nardi, D., and Patel-Schneider, P., *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, 2003.
7. Model theory is a formal semantic theory which relates expressions to interpretations. A "model" refers to a given logical "interpretation" or "world." (See, for example, the discussion of interpretation in Patrick Hayes, ed., 2004. *RDF Semantics – W3C Recommendation*, 10 February 2004.) The logic or inference system of classical model theory is monotonic. That is, it has the behavior that if S entails E then (S + T) entails E. In other words, adding information to some prior conditions or assertions cannot invalidate a valid entailment. The basic intuition of model-theoretic semantics is that asserting a statement makes a claim about the world: it is another way of saying that the world is, in fact, so arranged as to be an interpretation which makes the statement true. An assertion amounts to stating a constraint on the possible ways the world might be. In comparison, a non-monotonic logic system may include default reasoning, where one assumes a 'normal' general truth unless it is contradicted by more particular information (birds normally fly, but penguins don't fly); negation-by-failure, commonly assumed in logic programming systems, where one concludes, from a failure to prove a proposition, that the proposition is false; and implicit closed-world assumptions, often assumed in database applications, where one concludes from a lack of information about an entity in some corpus that the information is false (e.g., that if someone is not listed in an employee database, that he or she is not an employee.) See further, Non-monotonic Logic from the *Stanford Encyclopedia of Philosophy*.
8. Anon, "KBpedia - Open Standards," *KBpedia* Available: http://kbpedia.com/standards/.
9. Princeton University, "Wngloss(7wn) Manual Page," *WordNet 3.0 Reference Manual* Available: https://word-

net.princeton.edu/wordnet/man/wngloss.7WN.html.

10. See further CP 2.219-226 (1903). Also an earlier article that helps provide Peirce's views on communications is, "How to Make Our Ideas Clear."
11. Svenonius, E., *The Intellectual Foundation of Information Organization*, MIT Press, 2000.
12. Zeng, M. L., Žumer, M., and Salaba, A., *Functional Requirements for Subject Authority Data (FRSAD): A Conceptual Model*, Walter de Gruyter, 2011.
13. C.S. Peirce's sign relations are covered in the Representations section of *Chapter 2*. In the context of this discussion, the *sign* corresponds to any of the labels or identifiers associated with the (reference concept) *object*, the meaning of which is provided by its *interpretant*. See also John Sowa, 2000. "Ontology, Metadata, and Semiotics," presented at ICCS'2000 in Darmstadt, Germany, on August 14, 2000; see http://www.jfsowa.com/ontology/ontometa.htm.
14. A good explanation of this can be found in Rinke J. Hoekstra, 2009. *Ontology Representation: Design Patterns and Ontologies that Make Sense*, thesis for Faculty of Law, University of Amsterdam, *SIKS Dissertation Series No. 2009-15*, 9/18/2009. 241 pp. See http://dare.uva.nl/document/144859. In that, Hoekstra states (pp. 49-50): "RDFS has a non-fixed meta modelling architecture; it can have an infinite number of class layers because rdfs:Resource is both an instance and a super class of rdfs:Class, which makes rdfs:Resource a member of its own subset (Nejdl et al., 2000). All classes (including rdfs:Class itself) are instances of rdfs:Class, and every class is the set of its instances. There is no restriction on defining sub classes of rdfs:Class itself, nor on defining sub classes of instances of instances of rdfs:Class and so on. This is problematic as it leaves the door open to class definitions that lead to Russell's paradox (Pan and Horrocks, 2002). The Russell paradox follows from a comprehension principle built in early versions of set theory (Horrocks et al., 2003). This principle stated that a set can be constructed of the things that satisfy a formula with one free variable. In fact, it introduces the possibility of a set of all things that do not belong to itself . . . . In RDFS, the reserved properties rdfs:subClassOf, rdf:type, rdfs:domain and rdfs:range are used to define both the other RDFS modelling primitives themselves and the models expressed using these primitives. In other words, there is no distinction between the meta-level and the domain."
15. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S., *OWL 2 Web Ontology Language Primer*, 2012.
16. The canonical data model is especially prevalent in enterprise application integration. An entertaining animated visualization of the canonical data model may be found at http://soa-eda.blogspot.com/2008/03/canonical-data-model-visualized.html.
17. Minor exceptions, such as modeling recursion in ASN.1 (Abstract Syntax Notation), are so rarely encountered as to be dismissed.
18. As of the writing of this book, my census found hundreds of converters of various record and data structure types to RDF. These converters — also sometimes known as translators or 'RDFizers' — take some input data records with varying formats or serializations and convert them to a form of RDF serialization (such as RDF/XML or N3), often with some ontology matching or characterizations.