**Slide**

How many people in the room have ever played the game Guess who?  In the game you compete with an opponent to be the first to deduce which of 24 individuals appears on a card they’ve selected. You take turns asking yes or no questions - does your individual wear glasses? Does your individual have red hair? One strategy to deduce the individual on your opponent’s card more quickly is to start with the questions that will split the 24 in half. Let’s say one person in the deck wears a beret, so starting with the question does your individual wear a beret gives you just above a 4% chance of being correct, and if you’re incorrect, you can only remove one individual from the pile. However, if there is about an even split between brown eyes and other eye colors, then starting with the question ‘Does your individual have brown eyes” will inevitably split the pool in half.

This process of deducing to find the correct answer is a lot like designing an ontology in computer science. Computer ontologies are formal specifications that model the descriptive components of a knowledge domain.

**Slide**

So let’s say for each individual in the Guess Who knowledge domain, I were to attach as series of primitives:

Bernard is male.

… wears a hat.

… has brown eyes.

… has brown hair.

Anita is female.

… wears hair ribbons.

… has blond hair.

… has blue eyes.

**Slide**

Then I could much more readily create a search box to ask “present to me all of the individuals in Guess Who that are male, wear a hat, and have brown eyes.” The search box would return the following.

**Slide**

I could also make the ontology more complicated. ...perhaps saying that hair ribbons and hat belonged to the class “accessories.”

**Slide**

Then I could ask the system “present to me all of the individuals that are wearing accessories, and it would return all of the individuals that have been described as wearing hair ribbons or a hat. I could also add some logical primitives to the ontology to make my search box produce results even more quickly.

**Slide**

I could, for instance, say baldness and female – that is, they are two classifiers that will never overlap. Then when I run the search “present to me all the bald individuals” rather than sorting through all 24 responses, the computer could immediately segment out those labeled as female. This considerably speeds up search.

Building ontologies such as this is a key component of the field of knowledge representation. Knowledge representation is a subfield of artificial intelligence research concerned with getting computers to model, understand, and reason with natural language. Since the 1960s knowledge representation researchers have been theorizing how to model descriptive knowledge in order to build digital reasoning systems that can quickly filter information and make inferences based on partial knowledge inputs.

**Slide**

For example, in the early 1970s, Marvin Minsky, along with Seymour Papert, had attempted to tackle the challenges of teaching computers common sense knowledge through the construction of "micro-worlds" - or bounded environments with little complexity where computers could begin to learn. They would teach a robot to pick up a block for example by training the robot the difference between the concepts of “pick up” and “set down” and between “block” and “sphere.” Upon mastering knowledge in a micro-world, computers could slowly be introduced to more complex thinking. These systems generated a great deal of excitement in the field and demonstrated the promise of knowledge representation at the time. Yet in the mid to late 1970s, as AI researchers began to confront the inability of their systems to reason with more complex knowledge beyond the bounds of a constrained micro-world, funding for AI research began to dry up, marking the field’s first “Winter.”

**Slide**

As a highly bounded knowledge domain, Guess Who serves as micro-world. Of course, in the Guess Who example we run into the same issues that Minsky ran into as he sought to advance micro-worlds - that they are so reductionist that they inevitably break down as soon as they are exposed to more complexity. Because of course what counts as an *accessory* is open to interpretation, individuals identifying as many different genders can be bald, and we do not live in a world of all white Euro-American cartoon figures. Yet today, similar semantic technologies are foundational to the way that Web search gets carried out, to the way that Watson and other AI technologies make inferences, and to the way scientific researchers describe and share their data.

**Slide**

Ontologies such as this one attempt to encode the meaning and logic of descriptive elements in a data domain so that digital systems can interpret it and make decisions of it. In this talk I will refer to this as semantic infrastructure. This talk is about the cultural foundations of this meaning-making infrastructure. As a cultural anthropologist, I aim to characterize the diverse assumptions about language and diverse commitments towards the representation of difference that designers of semantic infrastructure bring to their work. I do so in order to better understand the design directives that have informed the technical configuration of semantic infrastructure – how ideas about language, difference, and complexity translate into and thus become interlaced in semantic infrastructure, in turn shaping how concepts get defined, how knowledge gets divided, and what can and cannot be represented in digital systems.

**Slide**

For the past four years, I have been ethnographically examining communities of researchers designing computer ontology languages, and thinking through best practices in digitally encoding and representing complex knowledge to computers and to other researchers. My primary fieldsites have been the semantic web community and the Research Data Alliance. In the Semantic web community, researchers and practitioners have been thinking through how to design languages for structuring, and logically modeling knowledge on the World Wide Web. The Research Data Alliance is a community of researchers and practitioners thinking through how to design the social and the technical infrastructure needed to advance international and interdisciplinary data sharing. This often involves conceptualizing and advocating for infrastructure that formalizes how research data gets described, interpreted, and modeled. I have conducted participant observation at conferences and workshops where this infrastructure is designed and debated; I have conducted a series of semi-structured ethnographic interviews with folks considered “experts” in each community, and I have traversed publicly accessible archives of design forums where many knowledge representation researchers congregate to discuss, debate, and design semantic infrastructure.

Through this research, I have come to learn that there is not just one unifying “culture of knowledge representation” but that there are diverse assumptions, commitments, methods, and design directives that designers of semantic infrastructure bring to their work. In this talk I’m going to unpack some of these cultures by describing how different semantic infrastructure designers have positioned themselves in relation to a key tension that emerges in their work (and arguably in language at large): the more they work to enable their systems to represent messy, fractured, “real world” knowledge, the less they are able to guarantee that their logic-based reasoning systems will always work and will always produce the “correct” answers. In other words, as they structure their knowledge representation languages in ways that allow users to “say anything about anything,” they usher in opportunities for unforeseen exceptions to the logical rules their languages encode. Thus knowledge representation designers have to decide how much to restrict what can be represented in logical languages vs. how expressive the languages can be.

**Slide**

In the mid-1940s, Alfred Tarski had been developing theories for determining the "truth" of any given sentence (or its formal semantics) by modeling the configuration of statements with formal logic. He proposed developing meta-languages, which would model "true" statements. For instance, one statement in a meta-language might be A satisfies (X and Y) if and only if ((A satisfies X) and (A satisfies Y)). Now, let’s fill the variables, A, X, and Y with ’Max, ’brother,’ and ’son’ to form the everyday-language statement, "Max is both a brother and a son if and only if Max is a brother and Max is a son." With this as a model, when the statement "Max is both a brother and a son" is true, a model theorist, using principles of formal logic, can then deduce further statements to be true, false, or ambiguous. For instance, the statement "Max is a brother," would be logically true; the statement, "Max is either a brother or a son," would be false, and the statement, "Max is a brother and/or a son," would be ambiguous. Tarski’s work would culminate with the emergence of model theory (which remains important to the design of the Semantic Web today).

**Slide**

At the time artificial intelligence was emerging as a discipline, the idea that language could be encoded via computerized algorithms was becoming increasingly viable. This would give rise to the study of knowledge representation - a sub-field within AI. Yet, John McCarthy and Marvin Minsky, often considered to be the "Fathers of AI," would eventually split considerably in their thinking about how best to approach representing knowledge to a computer.

In order to build machines that can replicate human thinking, memory, and language, a great deal of research throughout the "golden years" of AI (1950s to 1970s) had been oriented towards understanding, logically and algorithmically, how these human processes work. At the time, defense organizations like DARPA had been funneling money into the field, and as a result many digital systems were built to solve discrete problems. There was a great deal of optimism at the prospect of developing systems that could become just as intelligent as any human mind.

Almost paradoxically, knowledge representation researchers came to learn that the hardest challenges in their field were not getting computers to solve complex problems, but instead getting computers to solve very simple problems. Getting a computer to expertly play a game of chess turned out to be much easier than getting a computer to hold a coherent conversation for even just a few minutes. Thus, in the 1960s, research was reoriented to focus on designing systems that could explicitly exhibit common sense. One example is John McCarthy’s Advice Taker – a theoretical system that would apply first-order predicate logic to a series of declarative statements about situations. McCarthy’s idea was that any given situation could be adequately modeled for a computer to draw conclusions about declaratives. McCarthy provides the example of encoding the formal statements “I am at my desk” and “my desk is at my home” and the computer could then deduce that “I am at my home.” McCarthy had hoped that his colleague at MIT would help him build the Advice Taker, yet in the early 1970s, Minsky began advocating for a very different approach to advancing AI.

**Slide**

In his paper, a "Framework for Representing Knowledge," Minsky introduced the concept of the frame. For Minsky a frame is "a data structure for representing stereotyped information":

Here is the essence of the theory: When one encounters a new situation...one selects from memory a structure called a Frame. This is a remembered framework to be adapted to fit reality by changing details as necessary.

The "terminals" or slots of a frame have certain requirements that the values assigned to them must meet. In other words, in order to recognize and understand a situation, one must match the values of the situation to the terminals in various frames.

**Slide**

Suppose, for instance, you enter a room and notice a sofa and a television. You may first try to assign these items into the terminals for a living room frame.

**Slide**

But then you also notice bulletin boards, magazines, and a receptionist desk. These don’t fit into the terminals for the living room frame so you select a new frame - a waiting room area.

Applying this to AI involved creating "frame languages" based on descriptions of objects, (rather than algorithms for how data should be manipulated). As a machine is exposed to a new object, it compares and tries to match the object to the description for frames it holds in memory. If the object’s values can’t be assigned to the terminals of a frame, the machine has to select other frames from memory or "de-bug" existing frames to create new ones. Frame languages provided an alternative to reasoning with first-order predicate logic. While first-order predicate logic presumed to know what was universally true about things in the world ahead of time, frame languages could approach new and mundane situations, draw on prior understandings of the world, and adapt. In other words, for frame languages knowing how to manipulate data was more important than knowing what was true about the data.

**Slide**

Concerned that knowledge representation techniques at the time could not model broad knowledge, Minsky critiqued his own micro-world approach as he introduced the frame concept in 1974. Minsky argued that, while modeling logic in a micro-world often produced favorable results, "...as we approach reality the obstacles become overwhelming." Minsky also lamented that logic-based approaches to AI focused so intently on producing ’consistency and completeness’

I cannot state strongly enough my conviction that the preoccupation with Consistency, so valuable for Mathematical Logic, has been incredibly destructive to those working on models of mind. At the popular level it has produced a weird conception of the potential capabilities of machines in general. At the ’logical’ level it has blocked efforts to represent ordinary knowledge, by presenting an unreachable image of a corpus of context-free ’truths’ that can stand separately by themselves. This obsession has kept us from seeing that thinking begins with defective networks that are slowly (if ever) refined and updated.

Frames, on the other hand, focused, not on the internal structure of knowledge, but on how the mind came to recognize and structure external situations. While using first order logic to model knowledge assumed that concepts completely, consistently, and rationally followed the same set of rules, the architecture of the frame assumed that we can’t possibly model this way since every situation is marked with a new set of components or conditions. Minsky argued that our assessments and expectations of any given situation can never be more than imperfect approximations; we can only adapt a concept or situation from frames we already possess - to what have already been exposed to, or to what we have already experienced. They will thus never be complete or consistent.

**Slide**

Minsky’s frame-based approach to modeling knowledge along with others that were emerging in the early 1970s, such as Roger Schank and Robert Abelson’s 1975 concept of scripts, marked a schism in the community - one that Roger Schank coined as the "neats vs. the scruffies" in the early 1970s. Neats, working in the tradition of John McCarthy and Nils Nilsson tended to seek formal, clean, consistent, and complete solutions to AI problems. They believed that the world should be modeled with neat and well-defined semantics to correctly characterize the internal workings of a system. Scruffies, on the other hand, working in the tradition of researchers like Marvin Minsky and Roger Schank himself, asserted that the knowledge was too messy and illogical to model formally or consistently; they tended to employ hacks (typically through programming procedures, since hacks are often illogical) in their work to get systems to perform. For them, building computer intelligence was less about modeling the internal workings of a system correctly and more about creating pragmatic structures and protocols for roughly but rigorously assembling AI.

The neat-scruffy divisions in knowledge representation continued to be cited in journal articles, conference proceedings, and editorials through the early 1990s. The distinctions might be summarized with the following binary chart.

In the 1980s, Patrick J. Hayes advanced research in John McCarthy’s “neater” tradition as he introduced a naïve physics manifesto – a call for AI researchers to focus research on formally describing the world according to how regular people understand it (rather than as physicists understand it – as many “expert” AI systems had attempted to do). For instance, how do we formalize for a computer what it means to have a certain shape, what it means for something to be inside or outside, or what it means to exhibit force over another type of object?

**Slide**

Other researchers, such as Ronald Brachman and Hector Levesque, began to theorize and conceptualize knowledge representation systems that could formally describe the relationships between different entities in an information system. Their work emerged from Ross Quillian’s concept of semantic networks – a strategy for representing the relationships between entities in a domain by describing each entity and then linking the entities with predicates that formally described their relationships. Brachman took this one step further, arguing that in order to be able to make inferences from semantic networks, the logical relationships between entities and their classes needed to be more formally defined.

Brachman and colleagues aimed to build systems that could at once provide rich, or "expressive" descriptions of concepts, but could also make inferences about the relationships between concepts within a finite number of computational steps (or in other words, to build systems that were tractable). However, they began to theorize that the more expressive knowledge representation languages are, or the more complexity they attempt to model, the more challenging it is to guarantee the system will be sound, complete, and capable of producing tractable results.

**Slide**

For instance, what to do about “corner cases” – elephants are grey, except when they are white. Should the systems allow incomplete knowledge to be represented – for instance, should I be able to represent that John is not a student when I don’t know what John is. Should the systems be able to represent paradoxes, such as the ‘liar’ paradox? In order to guarantee completeness - to ensure the system was tractable, - folks in the neater tradition advocated for restricting the number of constructors, or "primitives," of the language to a small set. This would offer a substantially limited logic, in turn limiting the extent of "real world" complexity the system could represent. Folks in the scruffy tradition argued that these nice theoretical principles would break as soon as the models were exposed to “real world” knowledge.

**Slide**

Much of the research in knowledge representation at this time became about theorizing and testing how far they could push the tradeoffs between expressivity and tractability. Describing a knowledge representation system called CLASSIC that she worked on in the 1990s, Deborah McGuinness described in a 2016 interview with me:

The description logics have a benefit because they’re limited; they’re computationally tractable, but it’s almost impossible to use a limited language and meet all of your needs. So Classic had some extensions. [...] they had concessions to working in the real world. So we had these extra things that broke some of the nice properties [of] the theoreticians. In this part I wouldn’t be considered a theoretician because I was arguing for the usability features of it.

The fundamental differences in worldviews held by neat model theoreticians and scruffy system builders is perhaps best represented in a series of papers published in the Journal of Computational Intelligence in 1987. The instigating paper was Drew McDermott’s "A Critique of Pure Reason," aimed to tackle the "logicist" agenda put out by John McCarthy and Patrick Hayes. It was accompanied with 27 commentaries responding to his "critique."

The title of McDermott’s paper, of course, was a reference to philosopher Immanuel Kant’s book published under the same name in 1781. Although Kant was not cited anywhere in the paper, the influence of Kant’s thinking and work on the argument made in the paper is evident. Following Kant, McDermott aimed to critique assumptions about truth and deduction that had been guiding prominent work in computer science and knowledge representation for decades - most specifically, that truth could be formalized prior to its use or prior to experience of it.

For McDermott, the logicist agenda starts with the premise that knowledge should be formally represented by a series of logical axioms before the computer learns how to manipulate knowledge. He went on to argue that most common sense knowledge cannot be represented with logical axioms, and that instead AI researchers should be advancing systems that *know how* to do things with data rather than *know what* that data means/is. McDermott traced the lineage of this argument through several researchers - John McCarthy, Robert Moore, James Allen, Jerry Hobbes, and himself. However, in his paper, Patrick Hayes was positioned as the archetype of the logicist camp, and his naïve physics manifesto was the subject of the most scrutiny.

**Slide**

In Hayes’s response to McDermott, titled, "A Critique of Pure Treason," he wrote:

In the distant past, AI went through a bitter civil war between two camps, the Union of "neat logicists" and the "scruffy proceduralist" rebels. ... During the heat of the battle, the term "logic" came to have the emotive force of a war-cry for the neat faction, including myself, and was spoken by the proceduralists in the same sort of way that right- wing Republicans now speak of Communism, as an evil, cancerous blight whose spread should be resisted at all costs. This is now over, although the landscape bears scars, and it would be a shame if this paper were to start it up again. Unfortunately, terms such as "logicist" tend to blur hard-won distinctions.

**Slide**

Yet, in the same issue, Ronald Brachman, who had also been championing neater semantics for the past decade, lamented that the field had taken a "turn for the neat" - a "Great Formality Shift" that Hayes’s manifesto had heralded:

Back in the early days of the Shift, the impetus was (I think) to eliminate sloppy, inconsistent, and meaningless informal work by encouraging analysis in terms of a precise formal system. Yet, merely gentle suggestions that the tools of mathematical logic could be of use in AI ... have given way to dogmatic proclamations from the heart of the new Logicist camp: "There is only one language suitable for representing information - whether declarativist or procedural - and that is first-order predicate logic" (Kowalski 1980).

He went on to characterize what he called the "myth of one true logic," or Logicism "with a capital L." For Brachman, the myth of one true logic had become a form of "imperialism." Papers were not being accepted to conferences because they weren’t formal enough. "Hackers" were being shunned from the field. Systems based on anything but first order logic were not considered legitimate. Brachman’s assessment of the time was later captured in historical accounts of the field. In what is now the leading textbook on artificial intelligence, Stuart Russell and Peter Norvig, marked modern AI methodologies as leaning towards neatness:

Some have characterized this change as a victory of the neats - those who think that AI theories should be grounded in mathematical rigor - over the scruffies - those who would rather try out lots of ideas, write some programs, and then assess what seems to be working.

A few years further on, Pamela McCorduck (2004, 487) suggested, "As I write, AI enjoys a Neat hegemony, people who believe that machine intelligence, at least, is best expressed in logical, even mathematical terms."

**Slide**

Many in the semantic web community have described to me that a “semantic web” was Tim Berners-Lee’s vision for the World Wide Web from the very beginning – that we don’t produce much knowledge from hyperlinking web pages (as the web, as originally engineered, does). Such hyperlinking only tells us (and computers) that some piece of data on a web page is somehow related to another piece of data on another web page. Instead, the vision for the semantic web was to link data points within and between web pages, describing their relationships in languages that computers can interpret.

Berners-Lee made his first public reference to the semantic web during his talk at the first International World Wide Web conference, held at CERN in 1994. He wrote in his talk:

[based on hypertext] to a computer, then, the web is a flat, boring world devoid of meaning. [...] This is a pity, as in fact documents on the web describe real objects and imaginary concepts, and give particular relationships between them. [...] Adding semantics to the web involves two things: allowing documents which have information in machine-readable forms, and allowing links to be created with relationship values. Only when we have this extra level of semantics will we be able to use computer power to help us exploit the information to a greater extent than our own reading.

Slide

Today, semantic infrastructure serves as a foundation for many systems that organize, contextualize, and make automated decisions from data. For example, consider Google’s knowledge graph - a system that stores structured information on the Web into logically structured databases, so that you can search for “things not strings.” If I were to search for universities in Seattle in Google 15 years ago, the search engine would return pages that included the strings “universities,” “in,” and “Seattle.” somewhere in the text. Back then, search was based simply on matching keywords with strings on web pages. Since 2012, Google organizes certain “things” – like the University of Washington into ontologies – in this case “Seattle” and then “Colleges and Universities.” Data about University of Washington has been formally defined as a series of statements on the Web.

**Slide**

This is how I’m able to see the infobox to the right of the search results.

**Slide**

These statements are stored in graph databases where data points are individually linked to other data points or strings and the relationship between the two is described

**Slide**

with a property that has been defined on the Web.

Many different data specifications and standards have had to come together to make knowledge graphs (like Google’s and Facebook’s) possible. First, Semantic Web designers had to come up with a standard way to structure Web data into statements. The Resource Description Framework (or RDF) is a syntax for structuring data into triples – with a subject, predicate, and object. One piece of data serves as the subject; the object is another piece of data or a reference to something that exists in the world, and the predicate describes the relationship between the two data points.

**Slide**

They also needed to design schemas for formally defining what different predicates mean. Today, Schema.org acts as one such schema. Webmasters can tag content on their webpages with schema.org. For example “foundingDate” is a property defined in the Organization section of Schema.org, formally defined as follows: “The date that this organization was founded.” A webpage on the University of Washington may contain the RDF triple:

<div vocab=<http://schema.org/> typeof="EducationalOrganization">

<span property="name">University of Washington</span>

<span property="foundingDate" typeof="Date"> 1861<span>

</div>

The cast of characters building these knowledge representation tools for the World Wide Web overlap a great deal with those involved in the knowledge representation debates in the late 1900s, as well as their students. Drew McDermott, Patrick Hayes, and Deborah McGuinness, along with several other knowledge representation experts, all were involved in the design of semantic web technologies at various stages, and the arguments over neatness/scruffiness, formality/pragmatism, and tractability/expressivity continued to structure how they approached their work. Take, for instance, the Web Ontology Language.

The Web Ontology Language (OWL) is a language for modeling data. Using the language, webmasters can order their data into certain groups (or classes) and describe the relationships between data and classes. When asked questions about a dataset modeled with OWL, computers can follow the logic of the language to draw conclusions. Planning for the design of OWL took place in the early 2000s via a working group hosted by the W3C. With over 40 members, it was one of the largest working groups sponsored by the W3C and with that came very different understandings of how the ontology language should be structured. I analyzed the publicly accessible forums where the design of the ontology language was discussed as part of my fieldwork, and I interviewed several folks in the working group.

**Slide**

In a 2017 interview with Patrick Hayes, he argued that in the OWL working group he surprisingly played the role of the scruffy. When I asked if that role was comfortable for him, he said "It was then because I enjoyed making my fellow neats squirm. I think actually in Roger’s terminology, I started off as a neat, and I’ve gotten scruffier as I get older." I told him that he’s not the first person I’ve interviewed to tell me that. He laughed. "Yeah, well...your youthful idealism gets worn off by life when you get to my age."

**Slide**

In the working group for OWL, Patrick Hayes and several others had been calling for building an "expressive" logic - a logic that would allow researchers to "say anything about anything." In large part because of the differing worldviews that members of the OWL working group brought to the work, they ended up producing multiple versions of the ontology language – OWL-Full, OWL-DL and OWL-Lite. One of the primary differences between OWL-Full (the expressive version of OWL that enables webmasters to "say anything about anything") and OWL-DL (the restricted version of OWL that ensures that a reasoner can always produce an answer in a finite number of steps) is around the issue of meta-modeling, or creating models of models. OWL-Full allows meta-modeling; more specifically, it allows modelers to treat classes as instances of a meta-class. Let me explain. Classes typically serve as encompassing structures, and instances are members of those encompassing structures. You can compare this to the folder structure on a computer: folders (like classes) contain files (like instances). On a computer, you cannot treat a file as if it were a folder; files cannot contain anything else. The same is the case for OWL-DL; in OWL-DL, meta-classes are not allowed. Classes and instances are absolutely separate entities. So perhaps, I have a class called "endangered species," and it contains the instances "bald eagle" and "white leopard." In OWL-DL, since "bald eagle" is an instance, it cannot also be a class, and since "endangered species" is a class, it cannot also be an instance. Yet, in OWL-Full "bald eagle" can serve as both an instance and a class - perhaps an instance of the class "endangered species" and a class of individually named bald eagles. For a computer to be able to distinguish that "bald eagle" serves as both a class and an instance, a meta-model would need to be constructed. Meta-modeling adds layers of complexity to computer models. Restricting that complexity helps to guarantee that, when asked a question about some knowledge encoded in a knowledge base, a computer can produce an answer in a reasonable amount of time. In disallowing meta-classes, OWL-DL makes this guarantee. In allowing meta-classes, OWL-Full does not.

The proponents of OWL-Full had been arguing that the restrictions in OWL- DL would disallow webmasters from representing knowledge with greater complexity. Sorting knowledge into neat logical types may ensure that a computer can produce an answer in a reasonable amount of time, but it would preclude many knowledge orderings (including orderings that may produce paradox). To make OWL work for webmasters worldwide (webmasters that may order their data into diverse logical types) they argued that OWL needed to be looser, scruffier, and more tolerant of inconsistency. OWL needed to be able to model knowledge in a world riddled with paradox.

**Slide**

Thus, in an email thread with many working group members throwing around ideas for naming each version, Frank van Harmelen offered one possibility:

OWL Light: ?  
wimpy  
OWL fast: OWL/FOL-style Neat  
OWL large: OWL/RDF-style Scruffy 17

Similarly, reflecting on the battles that played out in the design of OWL, Guus Schreiber, co-chair of the OWL working group, summarized:

The debate about what a Web ontology language should look like is reminiscent of past neat-scruffy struggles. Knowledge modelers want expressiveness, logicians stress decidability [or tractability]. The main difference is that the Semantic Web actually forces us to make some choices: there is a strong need for real-world knowledge representation.

**Slide**

OWL, while still used for modeling the relationships between data in information systems today, failed to have a rapid uptake. Many Semantic Web designers have told me that early attempts at designing Semantic Web infrastructure failed to take off like the Web had in the early 1990s because they attempted formalize a set of rules (or in other words to standardize meaning) too soon. Echoing McDermott’s Critique of Pure Reason, Dame Wendy Hall described this in a June 2015 interview:

I talk about the Semantic Web going down an AI rat hole in that it got quite distracted by the AI community trying to sort out the issues before there was any data out there. [...] A lot of theory was talked about - a lot of upside down As and backward Es to try and prove things about ontologies and try and work out the theory of a Semantic Web before there was any data to experiment with.

Thus, as efforts to advance the semantic web progressed, instead of attempting to formalize logical rules ahead of time, the designers instead worked to build technologies for describing and ordering existing web data. Notably, data on the Web is scruffy, inconsistent, and full of paradoxes; it’s about as “real world” as you can get. David Woods used this image to describe to me the different perspectives around how to deal with modeling knowledge in this domain. To work in this domain the pure theorists constantly had to confront cases that violated their logical axioms.

Take for example, DBpedia, a project to extract information from Wikipedia pages and describe it with formally-defined schemas. Christian Bizer, co-founder of DBpedia narrated one of his “preferred examples” in a 2017 interview. He said. "Usually as a human, if I [ask]: Is a village and a tunnel the same? Or is a populated place and a tunnel the same? You would say no."

**Slide**

I confirmed. He noted: "It’s class disjointness." Class disjointness refers to a rule in many ontologies (including OWL) that specifies that two categories will never overlap, or that a data point will never be an instance of both categories. Marking disjointness restricts the ontology; it lets a reasoner know that it need not look through a class called populated place when asked a question about a tunnel. Because of this, specifying that classes are disjoint can make it easier for knowledge representation systems to infer relationships and produce results in a reasonable amount of time. In DBpedia, architectural structures are disjoint from populated places. Bizer continued:

A tunnel is not a populated place. [But] if you look at reality, or even if you look into Wikipedia, you find that there’s a tunnel in India that contains a slum, so a tunnel is a populated place. It violates your logical assumption, but still the logical assumption is quite useful. So if you want to cleanse Web data, even though it’s only 98% or 99% true, the class disjointness helps you. But there are cases, which are true which violate the axiom. So basically, I think the Semantic Web community thought for a long time that things would be easy, but now as we look at reality, as this example nicely illustrated, it turns out that things are not as easy as we hoped.

The “pragmatic” challenges of doing knowledge representation in the “real world” have been cited quite often in my fieldwork. The shift in semantic web efforts towards structuring data that was already out there provoked the community to revisit questions that had long troubled their work: how can we design ontologies that are complex enough to allow for diverse knowledge representation, restrictive enough to enable formal reasoning, and simple enough for everyday webmasters to understand?

**Slide**

Such questions have oriented the design of schema.org– an ontology developed as part of a collaboration between major search engines since 2011. I have often been told that despite schema.org’s lack of logical rigor, it is “winning out” in the semantic web domain – with a viral growth that other semantic web technologies never achieved.

There is general consensus in the schema.org community that the schema should not become an “ontology of everything,” listing and defining every classifier that may be used to characterize web content. Instead, the schema.org community has aimed to build out a “middle ontology” – one that most webmasters will find accessible and useful (Ronallo 2012). This has meant policing what can and cannot be represented in the schema.

The creators of Schema.org were incredulous towards knowledge representation efforts that attempted to formalize ontologies before there was data to experiment with. Guha et al. (2016, 50), the creators of Schema.org describe:

Schema.org also shares the linked-data community’s skepticism toward the premature formalism (rule systems, description logics, and so on) found in much of the academic work that is carried out under the Semantic Web banner.

Skeptical of premature formalism, Schema.org directors write in their wiki that an overarching design approach for the schema is to remain flat - to avoid nesting concepts and modeling with hierarchy: "there is no single right way to model anything. For our purposes, we have a bias towards flat models" (W3c, 2012). However, without hierarchies inconsistencies could be introduced into the system of knowledge representation.

**Slide**

Consider one issue that emerged as the Schema.org community attempted to develop a property for the concept ’abdomen’. Abdomen can at once refer to a body part and also refer to a type of physical medical exam. In a flat schema, however, there is no way of disambiguating these two uses of the same word.

**Slide**

Disambiguating them would require finer levels of granularity - noting that ’abdomen’ meant body part in this context, and medical exam in another. Yet imposing such levels would stratify a flat ontology. In other words, the Schema.org community has to work at a tricky limit: they must define the infrastructure "flatly" in order to enable concepts to take on new meaning as they move in diverse and unpredictable settings, yet in doing so, they enable concepts to take on a surplus of meaning (or to be polysemic) in any given setting - at times in instances when there is a difference that makes a difference between two uses of a word.

**Slide**

Referencing this issue, one designer responded to calls for neater semantics to sort out the abdomen inconsistency:

I think you are rightly pointing to the implicit initial (and naïve) assumption of schema.org, which is that the whole world can be represented under a single flat namespace at arbitrary level of granularity, with natural language words as identifiers. Obviously, this does not scale and hits quickly the wall of polysemy, as the Abdomen example perfectly illustrates, and we are bound to have more of the same with the schema growth (which is, remind you, potentially unbound[...])

That said...other problems you point at (lack of documentation, semantic glitches etc) will always be present in this scruffy-work-in-progress called "Web semantics" (read: fuzzy, plural, inconsistent etc). I’m sure you will ever ever fight it with all your will and strength given where you come from, but I’m afraid this battle has been lost for quite a while now. As Pat Hayes told me a while ago "My ivory tower has been seriously shaken these days, waters of real world are slowly rising around us." Time to learn swimming in troubled waters ...

**Slide**

Towards the middle of Peter Norvig’s keynote at the WWW2016 conference, he put up a slide that presented two books - the first Ronald Brachman and Hector Levesque’s Knowledge Representation and Reasoning, the second Vinit Nayak’s Copying and Pasting From Stack Overflow. With a click of the mouse, a large red prohibition symbol displayed over Brachman and Levesque’s book. For Norvig, meaning did not come from highly trained logicians carefully building out ontologies but instead from scruffy hackers, marking up their data with available syntax. His talk did not reflect a "victory of the neats" or a "neat hegemony," as the field had been characterized (in his book) a decade before. Instead for Norvig, there was too much ambiguity on the Web. At the end of Norvig’s talk, an audience member asked him at what point logicians should be re-enlisted in the knowledge representation project. Norvig replied that he believed the Semantic Web of the future would open a job market for philosophers.

**Slide**

He went on to say, "Maybe the real world is less real than we think [...] It’s not so much about ontology. It’s more about epistemology."

The Web, as an information infrastructure, has come to be seen as a "real world" context for testing ideas in knowledge representation. People from all over the world contribute to the Web, so its data is scruffy, inconsistent, and ambiguous.

**Slide**

How to model knowledge in a decentralized information space like the Web- a space where overnight Paul Ryan shows up on the Wikipedia page for invertebrates – a move certainly to introduce errors into knowledge systems that assume that the categories of invertebrates and humans never overlap and that keep politics and nature divided. How to formally define words like “woman” or “worker” -words that Spivak argues are both absolutely necessary and absolutely violent to define. Knowledge representation experts in AI have been confronting these epistemological challenges as they work in messy spaces like the WWW. I have consistently heard them refer to this as challenges that emerge in the real world or “in the wild” – outside of the carefully curated microworlds where knowledge is defined by the specialist and meaning can be policed or controlled.

**Slide**

Semantic infrastructures impact how data is ordered, described, and presented to us. Schema.org impacts what appears in search results. It is being recommended to scientists interested in research data sharing. RDF is used to structure open government data, as well as scientific data. OWL is used in healthcare analytics, and other semantic technologies are being used in IBM’s Watson and Amazon’s Alexa. Their epistemological underpinnings matter for how we access data – whether what can be represented is deliberately restricted to get information systems to produce an answer and to control how data can be interpreted – which notably can be important for sharing data across disciplines. or whether it has been designed to be expressive – offering flexibility in representation but also giving up the guarantees that there will be systems can produce an answer around that data and that diverse communities will know unambiguously the context in which a certain data point is being used.

We need more research into how knowledge is ordered, disseminated, and eclipsed as “knowledge experts” confront epistemology ontologically and ontology epistemologically – words that themselves are scruffy, inconsistent and founded on pluralized logics.