Knowledge Representation, the World Wide Web, and the Evolution of Logic[*]

Christopher Menzel
Department of Philosophy
Texas A&M University
College Station, TX 77843
cmenzel@tamu.edu

Computer networks are nearly as old as computers themselves. However, prior to the Internet — the infrastructure of the Web — such networks were all closed: only designated computers — within the same business or organization — were able to join and, typically, each network had strict control over the sorts of information would be exchanged on the network and in what forms. Perhaps the most significant feature of the Web is that it is *open*, indeed radically so. Unlike closed networks, the Internet is accessible to a huge segment of the world's population, and to publish virtually any content, one needs only access to a web server.

From its earliest days, it was recognized that the radical openness of the Web presented a revolutionary way of thinking about information. Typically, in closed networks, the information that is available is closely controlled and monitored by the administrators of the network. The Web, by contrast, is "anarchic"; it has no administrators. Because there are so few hindrances to the publication of content, new information is constantly appearing on the Web, from the lightest entertainment news to cutting edge results in science, engineering, and medicine; from political rhetoric to pornography.

Human users thrive in such a milieu; but the Web, like other computer networks, is also a medium of exchange for formalized information intended for use by software agents, with little or no human intervention. The question then becomes: How can formalized information be represented on the Web in such a way that it can be identified by interested agents (notably, software agents), reasoned upon, and integrated with related information? The problem of identifying relevant information is essentially the problem of *search* technology, and that continues to be addressed effectively by, most notably, Google. Suitable *representation* of information is the more challenging problem.

[*] My sincerest thanks to my friend and collaborator Patrick Hayes for countless discussions over several years about the issues the led to the development of Common Logic and also for his help with this paper.

To address this challenge, many researchers have turned to first-order logic (FOL). And not without good reason. In most philosophical contexts for the last sixty-odd years at least, FOL of some ilk, perhaps with modal or other intensional operators, has been the chief medium of formal representation. This is of course a very good thing. FOL, with its clean, well-understood syntax and semantics, allows for the clear expression of philosophical ideas and their logical connections. Indeed, an argument or philosophical theory rendered in FOL is perhaps the cleanest example there is of "representing philosophy".

Modern FOL (as well as higher-order logic) of course is typically traced back to Frege. Interestingly, even though Frege's metaphysical and semantical views are in many respects out of favor, a number of prominent syntactic and semantical properties of FOL still reflect distinctly Fregean ideas, notably, that there is an inviolable divide between concept and object, between the meanings of predicates and the meanings of terms. Indeed, FOL seems to generalize this metaphysical division by segregating objects from functions from properties from relations and, moreover segregating relations and functions internally according to arity.

These divisions, taken at face value, embody a significant metaphysical viewpoint, one that can in fact hinder or prejudice the representation of philosophical ideas and arguments. Some philosophers have of course noticed this and have, accordingly, sought to alter or extend traditional FOL in novel ways to accommodate more flexible and egalitarian metaphysical frameworks. My primary purpose in this paper, however, is to document and discuss how similar changes to FOL have been motivated by its more practical and applied encounter with the problem of representing, sharing, and reasoning upon information on Web. More specifically, following a bit of relevant history, in the following sections I will describe four "evolutionary adaptations" of traditional first-order logic that were motivated initially by developments in knowledge representation and the Web and more recently, and more urgently, by the rise of the so-called *Semantic Web* — adaptations that concern representational features that stem largely from its Fregean heritage. These developments have culminated in a standardized logical framework known as *Common Logic* — described in the penultimate section of the paper — in which that the rigid metaphysical divisions of traditional FOL are largely eliminated in favor of a more egalitarian metaphysics that is not only better suited to the needs of the web but also (I will briefly suggest) a more faithful rendering of the idea that logic should be "topic neutral".

1 Salient Features of Traditional First-order Logic

Because we will be discussing a certain evolution in FOL, it will be useful to begin with a brief summary several of its familiar features that are particularly important for purposes here.

1.1 Syntactic Features

- **A tripartite lexicon** Over and above the usual array of logical constants (Boolean operators, quantifiers, and identity) and delimiters found in a (typical) traditional first-order language $\mathscr{L}$, the class of primitive, nonlogical syntactic elements — collectively, the *lexicon* — of $\mathscr{L}$ comprises three categories including: (individual) constants (*Cn*), function symbols (*Fn*), and predicates (*Pr*).

- **Fixed syntactic adicity** Every function symbol and every predicate has a fixed, finite *adicity* that determines the number of arguments it can take to form a legitimate function term or atomic formula. Because constants can take no arguments, they have no adicity.[1] An n-adic function symbol or predicate is also said to be an *n-place* function symbol or predicate.

- **Strict syntactic typing** No member of any syntactic category can play the syntactic role of the members of any other category. Specifically:

  - In an atomic formula, among lexical items, only an individual constant or function term can occur in argument position and only a predicate can occur in predicate position. In particular, a predicate cannot be predicated of itself.

  - In a function term, among lexical items, only an individual constant or function term can occur in argument position and only a function symbol can occur in function position. In particular, a function symbol cannot be applied to itself.

- **No predicate quantifiers** Variables can occur only where individual constants can occur in atomic formulas. Hence, in particular, *quantified* variables can occur only in argument position.

1.2 Semantic Features

Corresponding to these syntactic features are familiar semantic features of first-order interpretations:

- **A tripartite ontology** Every interpretation for a first-order language $\mathscr{L}$ consists of three semantic classes: a set D of individuals, a set of functions over D, and a set of relations over D to serve as the semantic values of the individual constants, function symbols, and predicates, respectively, of $\mathscr{L}$.

- **Fixed semantic adicity** Every function and every relation has a fixed, finite adicity that determines the number of arguments it can apply to. (In particular, properties are simply 1-place relations.) n-adic function symbols signify n-adic (or n-place) functions on the domain of individuals, n-adic predicates signify n-adic relations. Individuals, having no functional or predicative "nature", have no corresponding adicity.

- **Strict semantic typing** In an interpretation of a first-order language, the semantic values of constants ("individuals") are of an entirely different type than the semantic values of function symbols and predicates.[2] Consequently, no member of any semantic category can play the semantic role of any other category. Specifically:

    - Relations can only be exemplified by individuals (in particular, a property cannot exemplified itself); only relations can be exemplified.

    - Functions can only apply to individuals (in particular, a function cannot be applied to itself); only functions can be applied to anything.

- **No quantification over "higher order" entities** Quantifiers range only over individuals. Hence, given that semantic categories are disjoint, properties, relations, and functions lie entirely outside the range of the quantifiers. One must turn to second-order logic to quantify over relations or functions.

Two additional semantic features characterize traditional first-order interpretations:

- **Extensionality** Functions and relations are understood *extensionally*. That is, functions are identical if they map the same arguments to the same values and relations are identical if they are true of the same objects. Typically, extensionality for functions and relations is ensured by defining them to be *sets*; an n-place relation is simply a set of n-tuples and an n-place function is simply a set of n+1-tuples satisfying a certain "functionality" condition that ensures that every function yields a unique value for any given argument.[3]

- **Variable assignments** Variables are assigned individuals relative to a fixed interpretation for the constants, function symbols, and predicates. Truth is defined in terms of variable assignments.

In the evolution of logic described in this paper, all of these features, syntactic and semantic, disappear.

1.3 Interpretations, Satisfaction, and Truth

For the sake of clarity, we present a relatively standard notion of an interpretation for first-order languages which will be modified incrementally in accordance with corresponding modifications to the notion of a traditional first-order language. Since the culmination of an interpretation is a definition of truth for formulas, it is useful to add to the definition of a traditional first-order language $\mathscr{L}$ in §1.1 the definitions of terms and formula of $\mathscr{L}$. (Quotation will only be used when it seems particularly helpful.)

1. Every individual constant and variable (of $\mathscr{L}$) is a *term* (of $\mathscr{L}$).

2. If $\tau_1, \ldots, \tau_n$ are terms and α is an n-place function symbol, then $\alpha(\tau_1,\ldots,\tau_n)$ is a term.

3. If $\tau_1, \ldots, \tau_n$ are terms and π is an n-place predicate, then $\pi(\tau_1,\ldots,\tau_n)$ is an (*atomic*) *formula.*

4. If φ and ψ are formulas, then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \supset \psi)$, and $(\varphi \equiv \psi)$ are formulas.

5. If φ is a formula and ν a variable, then $\forall\nu\varphi$ and $\exists\nu\varphi$ are formulas.

6. Nothing else is a formula.

Following common practice, all quantifiers except the first in a string of similar quantifiers will be be dropped, i.e., $\ulcorner\forall\nu_1\ldots\nu_n\varphi\urcorner$ and $\ulcorner\exists\nu_1 \ldots\nu_n\varphi\urcorner$ will be written in place of $\ulcorner\forall\nu_1\ldots\forall\nu_n\varphi\urcorner$ and $\ulcorner\exists\nu_1\ldots\exists\nu_n\varphi\urcorner$.

An *interpretation* for a first-order language $\mathscr{L}$ is a 4-tuple $\mathbf{I} = \langle D,den,fext,rext\rangle$, where D is, intuitively, the set of *individuals* of $\mathbf{I}$ and *den*, *fext*, and *rext* are functions that map the basic syntactic items in the lexicon of $\mathscr{L}$ to appropriate semantic values: specifically, *den* maps the constants Cn of $\mathscr{L}$ to individuals of **I**; *fext* maps the function symbols Fn of $\mathscr{L}$ to (extensional) functions from individuals to individuals; and *rext* maps the predicates Pr of $\mathscr{L}$ to (extensional) properties of, and relations among, individuals. A little more exactly:

- $den : Cn \longrightarrow D$;
- $fext : Fn \longrightarrow \{f \mid f : D^n \longrightarrow D$, for some $n \in \mathbb{N}^+\}$[4];
- $rext : Pr \longrightarrow \{r \mid r \subseteq D^n$, for some $n \in \mathbb{N}^+\}$.

rext and *fext* are of course so constrained that the adicity of a predicate or function symbol must be identical to the adicity of its semantic value. Given an interpretation **I** of $\mathscr{L}$, denotations for the terms of $\mathscr{L}$, and satisfaction for the sentences of $\mathscr{L}$ in **I**, can be recursively defined in the familiar sort of way. Specifically, for a variable assignment s, let s' be the usual extension[5] of s to all the terms of $\mathscr{L}$ and, for variables ν, say that t is a *ν-variant* of s if $t(\mu) = s(\mu)$, for all variables $\mu \neq \nu$. Then, for a given variable assignment s:

- s satisfies $\pi(\tau_1,\ldots,\tau_n)$ in **I** if and only if $\langle s'(\tau_1), \ldots, s'(\tau_n)\rangle \in rext(\pi)$
- s satisfies $\neg\varphi$ in I if and only if s does not satisfy φ; similarly for the other Boolean cases.
- s satisfies $\forall\nu\varphi$ ($\exists\nu\varphi$) in **I** if and only every (some) ν-variant t of s satisfies φ in **I**.

Truth in **I** is then defined in the usual way: A sentence φ of $\mathscr{L}$ is *true* in **I** if every variable assignment satisfies φ. **I** is said to be a *model* of set S of sentences of $\mathscr{L}$ if every sentence in S is true in **I**.

2 Knowledge Representation and Open Networks

2.1 Knowledge Sharing and the O(n^2) Problem

Long before the rise of the Web it was recognized in the Artificial Intelligence (AI) community that well-defined, logic-based languages are extremely effective frameworks for representing information clearly, and in a way that is subject to machine processing. This recognition gave rise to the development of a wide variety of knowledge representation (KR) frameworks built upon first-order logic or some fragment thereof, for example, SNePS (Shapiro 2000), LOOM (MacGregor & Bates 1987), CLASSIC (Patel-Schneider et al. 1991), Ontolingua (Gruber 1992), CycL (Lenat & Guha 1991) not to mention any number of lesser known in-house systems. Because they are all logic-based, information represented in any such system possessed the clarity and rigor that is needed to ensure understanding by human agents, to enable the use of automated reasoning techniques to draw useful inferences from the information, and to integrate that information with related bodies of information — *so long as that information was also expressed in that system's own notation*.

With the rise of these logic-based systems, the latter limitation became acute. If I am working on, say, a diabetes knowledge base in LOOM, and you already have an extensive knowledge base on the production of insulin by the pancreas in Ontolingua, I must have some sort of translator to be able to get your information into my system. And you'll need one to get my information into yours. This leads to what has been called the "order *n*-squared" ($O(n^2)$) problem. Suppose you wish to establish a collaborative environment with, say, three other research groups, each of which uses a different KR system. To exchange information between these systems, twelve (4^2 - 4) translators will be needed, two for each pair of distinct systems. If you then wish add a further group into the environment that is using yet another system, then you will be forced to find or develop eight new translators for a total of twenty ($5^2 – 5$), as depicted in Figure 1. More generally, then, in an environment with *n* distinct systems, the inclusion of a further system requires 2*n* new translators to enable complete communication between systems.

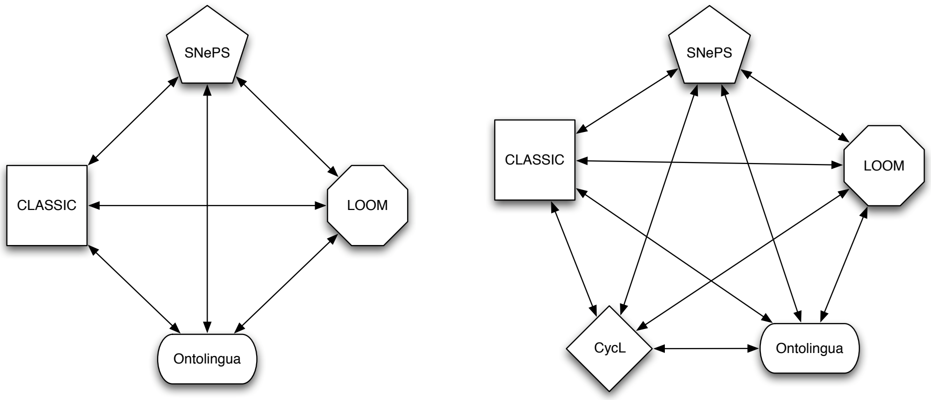

Figure 1: The $O(n^2)$ problem

The desire for a more efficient solution to the $O(n^2)$ problem than the construction of piecemeal translators (among other motivations) led to the development of the

Knowledge Interchange Format, or KIF, largely the work of Stanford University computer scientist Michael Genesereth (1998). The idea behind KIF was simple: If there were a single "interlingua", i.e., logic-based language capable of expressing everything that is expressible in any of the systems in a given environment, then, to exchange information between systems, instead of O(n^2) pairwise translators, one would only need O(n); specifically: for each system S, a translator from S to KIF and another from KIF to S. Thus, the addition of a new KR system into an environment of n systems + KIF requires only two new translators instead of 2n, as depicted in Figure 2.

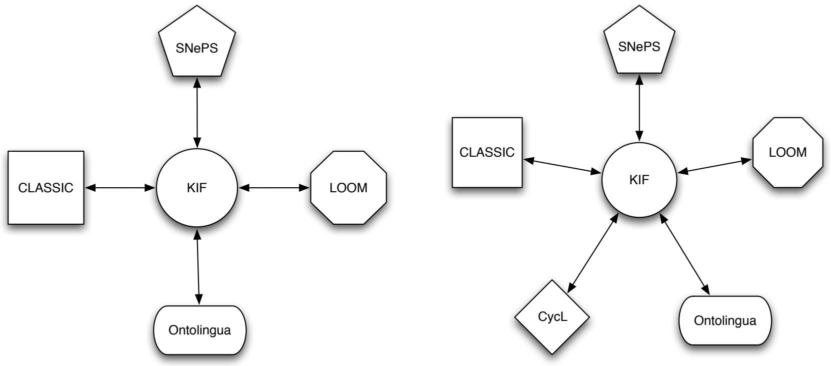

Figure 2: The KIF "Interlingua" solution to the O(n^2) problem

2.2 Pull Technology and the Rise of the Open Networks

However, with the rise of open networks the scenario that gave rise to the O(n^2) problem began to appear increasingly "academic" and artificial, most notably in the context of the so-called *Semantic Web*. The goal of the Semantic Web (a project involving many hundreds of people around the world) is to exploit the information-interchange protocols of the largest and most open of networks — the World Wide Web — in a more systematic and standardized way by having software systems exchange machine-readable information expressed in one or more logic-based representational frameworks. The O(n^2) problem looks artificial in the context of the Semantic Web simply because it is essential to the architecture of the Web that it makes no presuppositions about the existence of the sorts of two-way connections that could, in the event of unclarity, disagreements, or other failures or disruptions of communication, be used to facilitate negotiations and agreements about content and its representation. Rather, the default presupposition of the Web is that there simply are no such connections and, indeed, that they might not even be possible.

The reason for this is that, in regard to knowledge sharing, the Web is primarily a “pull” technology: users get only what they go out and actively retrieve. There is no easy way, and perhaps no way at all, within this architecture, to set up a two-way communication path between the composer of the information and the user of it; a path that could be used to negotiate an agreement. A basic fact about publishing information on the Web is that publisher has no control over, or even knowledge of, how, where, and how much of the published information will be used. The architectural model of the Web allows for content to be archived, for example, introducing arbitrary delays between publication and consumption, while mandating that the meaning of the archived information should not be lost. Thus, there is no way for a publisher and a consumer of some piece of information to come to any further agreements about mutually accepted notational conventions, which vocabulary to use, and so on. Clearly, however, the meaning intended by the publisher should be retained by the consumer to the fullest possible extent. Thus, just as one must use HTML in order for one's published website to be seen by others, publishers of web content must all cleave to a set of universally accepted logical conventions if their content is to be *understood* by others: a universally accepted standard language with a clearly defined syntax and a universally accepted semantics in the form of a model theory.

Not just any semantics will do, however. In order not to limit the scope of publishable content, the semantics should not place any constraints on the expressibility of the standard language. Moreover, the semantics should be *monotonic*. Pull technology requires that published content *C* remain stable in all contexts. Hence, it is critical that the semantics of the language in which *C* is expressed guarantee that the addition of new information *C** to *C* have no effect on the logical properties of *C* itself. New content might of course *contradict C*, but it should not *alter* it — anything that one could infer from *C* prior to the addition of *C** should still be inferable after it is added. This is precisely the property of monotonicity.

These consideration, suggest that a standardized framework for publishing content on the web should be based upon full first-order logic. Not only is the syntax of FOL clear and standardizable and its semantics monotonic, but FOL is generally considered the benchmark for expressibility — it is at least as expressive as any standard KR language, but not so expressive that logical consequence fails to be recursively axiomatizable.

2.2.1 KIF Again

Although unneeded after all as an interlingua for solving the O(n^2) problem, KIF is based upon first-order logic and hence seems to be viable, standardizable framework for publishing Web content. Moreover, KIF's actual syntax is especially conducive to use on the Internet, and is indeed one reason why it in fact has become widely used as a medium for expressing content among KR practitioners. KIF's syntax was inspired by the programming language LISP — famous for its economical and elegantly simple syntax — and was designed so that all KIF expressions consist of ASCII characters; notably, instead of the usual quantifiers and connectives of *Principia*-style languages — $\forall$, $\exists$, $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$ — KIF used `forall`, `exist`, `not`, `and`, `or`, `=>`, and `<=>`. Also, like LISP, KIF adopted prefix rather than infix notation for its binary propositional operators. So, for example, *Every boy kissed a girl* in traditional syntax would be represented as

(1) $\forall x(\mathrm{Boy}(x) \supset \exists y(\mathrm{Girl}(y) \wedge \mathrm{Kissed}(x,y)))$.

In KIF, by contrast, this is expressed as

(2)
```
(forall (x)
          (=> (Boy x)
               (exists (y)
                          (and (Girl y) (Kissed x y))))) .
```

In doing so, KIF reflects the shift from pen, paper, and, ultimately, typeset publications as the primary media for authoring and communicating logical information to computer keyboards and computer networks via electronic protocols like email and mailing lists. It also represents an important start in a series a significant developments in the language and semantics of FOL motivated by the development of open networks.

3 Four Evolutionary Adaptations of FOL

The shift to an ASCII syntax was the least significant change influenced by the rise of computer and network technology. However, KIF also anticipated the first of four more significant "adaptations" of traditional FOL that were motivated by pressures arising from the development of the open networks.

3.1 Adaptation I: Variable Polyadicity

In addition to the more superficial departure from traditional first-order syntax just noted, KIF also abandoned two of the salient features of traditional first-order logic noted above — viz., syntactic and semantic adicity — in order more effectively to integrate information found across different knowledge bases. Neither the predicates and function symbols of KIF nor their semantic counterparts have a fixed adicity. What this means semantically is that, in KIF, a relation can consist of ordered tuples of arbitrary finite length; similarly for functions, subject as before to an appropriate functionality condition to ensure a unique value for every argument sequence.[6]

The reason for this adaptation is simply that assumptions regarding the number of arguments that a property, relation, or function can take can vary widely from context to context. As a simple example, in one knowledge base, 'Teacher' might have only one argument, e.g.,

(3) Teacher(Plato)

while in another it might take two:

(4) Teacher(Plato,Aristotle)

and in yet another — one that includes a time interval parameter indicating the period over which the *Teacher* relation holds between the first two arguments — three:

(5) Teacher(Plato,Aristotle,364-360BCE)

In traditional FOL, each of these contexts would have to be represented by a distinct predicate with special axioms relating that predicate to the others. Intuitively, however, there is simply one predicate capable of taking varying numbers of arguments depending on how many explicit qualifying "roles" one wishes to include in a statement about Plato's teaching career. KIF, with its "variably polyadic" syntax, the same predicate can serve for all three contexts.

Syntactically, then, to accommodate variable polyadicity more formally, the adicity condition on function symbols and predicates is removed and the grammar is modified accordingly. More exactly, the language $\mathscr{L}_1$ is defined to be the result of modifying the traditional first-order language $\mathscr{L}$ such that:

1. The members of *Fn* and *Pr* are not assigned a specific adicity;
2. If $\tau_1, \ldots, \tau_n$ are terms and $\alpha \in Fn$, then $\alpha(\tau_1, \ldots, \tau_n)$ is a term of $\mathscr{L}_1$; and
3. If $\tau_1, \ldots, \tau_n$ are terms and $\pi \in Pr$, then $\pi(\tau_1, \ldots, \tau_n)$ is an atomic formula of $\mathscr{L}_1$.

Semantically, the shift to variable polyadicity requires only that, in the definition of an interpretation above, (i) the now inapplicable constraint on *fext* and *rext* that the adicity of a predicate or function symbol must be identical to the adicity of its semantic value is removed and (ii) the ranges of their possible semantic values is broadened appropriately. Specifically, where D^* is the set $\bigcup_{n \in \mathbb{N}^+} D^n$ of all finite sequences (i.e., *n*-tuples, for arbitrary *n*) of members of *D*:

- $fext : Fn \longrightarrow \{f \mid f : D^* \longrightarrow D\}$
- $rext : Pr \longrightarrow \{r \mid r \subseteq D^*\}$.

On this definition, the functions that interpret function symbols, and the relations that interpret predicates, are now themselves variably polyadic; that is, the functions in question are now total functions on the set of all finite sequences of individuals in the domain of the interpretation and the relations in question can now contain sequences of varying finite length. Notice that this change *simplifies* the descriptions both of the syntax and the truth conditions of the language; this kind of simplification by removal of constraints will be typical of the evolutionary changes we will describe.

Variable polyadicity was not new with KIF. Kenny (1963), in particular, suggested it as the natural syntax for action sentences and a formal syntax and semantics of variably polyadic predicates was explicitly developed by Grandy (1976).[7] However, there is no indication that Genesereth or any of other contributors to KIF were aware of any such work. Rather, along with its adoption of an ASCII-based syntax, KIF's variable polyadicity was an adaptation motivated entirely by its intended role as a general interlingua for facilitating communication between networked knowledge bases using distinct KR frameworks.

3.1.1 Sequence Variables: A Useful Complication

KIF also introduced an novel extension to conventional FOL syntax, one which is a pragmatic necessity when predicates are allowed to become polyadic. Consider as an illustration the polyadic equality relation AllEqual, which is true when all its arguments are identical, for any number of arguments. How can this be described using first-order axioms? The case of two arguments can of course be expressed as

(6) $\forall xy(\mathrm{AllEqual}(x,y) \equiv x=y)$

but this only handles one of the denumerably many possible cases. To handle the rest, an *infinite* set of axioms is needed:

(7) ∀xyz(AllEqual(x,y,z) ≡ (x=y) ∧ (x=z))

(8) ∀xyzu(AllEqual(x,y,z,u) ≡ (x=y) ∧ (x=z) ∧ (x=u))

and so on. Axiom sets which assign intended meanings to variably polyadic predicates are almost always infinite in this way, and this poses a problem for those who seek to capture such meanings in a finite document. The KIF solution was to provide a new quantifier pattern which can, in effect, summarize infinite sets like this. This is done by adding a new category of name, which, in KIF, are called *sequence variables*. Formally, a sequence variable is simply a variable of a special distinguished kind that can be bound by a quantifier in the usual way, and can occur as an argument.[8] KIF wrote these prefixed by the symbol '@', but for purposes here 's', 's_1', 's_2', … will simply be reserved for sequence variables.

Semantically, given an interpretation **I** for a language with sequence variables, a variable assignment *s* assigns to each such variable a *finite sequence* of individuals in the domain *D* of **I**. The sequence formed by interpreting the various arguments of a function term and concatenating the results in order is the sequence that the corresponding function operates upon; similarly for predicates. More exactly, for individuals *a* and *b* and sequences $S = \langle c_1,\ldots,c_n\rangle$ and $S' = \langle d_1,\ldots,d_m\rangle$ of individuals, let $a^\frown b = \langle a,b\rangle$, $a^\frown S = \langle a,c_1,\ldots,c_n\rangle$, $S^\frown a = \langle c_1,\ldots,c_n,a\rangle$, and $S^\frown S' = \langle c_1,\ldots,c_n,d_1,\ldots,d_m\rangle$. Then, for function terms α, predicates π, and terms τ_1, …, τ_n:

- $s'(\ulcorner\alpha(\tau_1,\ldots,\tau_n)\urcorner) = fext(\alpha)(s'(\tau_1)^\frown\ldots^\frown s'(\tau_n))$.

- *s* satisfies $\ulcorner\pi(\tau_1,\ldots,\tau_n)\urcorner$ in **I** if and only if $s'(\tau_1)^\frown\ldots^\frown s'(\tau_n) \in rext(\pi)$.[9]

This additional machinery allows infinite collections of axioms that conform to regular recursive patterns to be described as one or two sentences. Our example above can be axiomatized fully by two simple sentences:

(9) ∀x(AllEqual(x)

(10) ∀s∀x∀y (AllEqual(x,y,s) ≡ ((x=y) ∧ AllEqual(x,s)))

For example, by (10), where 'x' is instantiated to 'a', 'y' to 'b' and 's' to 'c,d',

(11) AllEqual(a,b,c,d)

is equivalent to

(12) $a=b \wedge \mathrm{AllEqual}(a,c,d)$.

By the same token, the second conjunct of (12) is equivalent to

(13) $a=c \wedge \mathrm{AllEqual}(a,d)$,

whose second conjunct is equivalent to

(14) $a=d \wedge \mathrm{AllEqual}(a)$

Putting these together, given both the axioms (9) and (10), (11) unpacks ultimately to

(15) $a=b \wedge (a=c \wedge a=d)$

as desired. This kind of recursive expansion to a “terminal” case is typical of the axiomatic style one learns to use when writing axioms with sequence variables. Programmers will note that it is closely similar to the style of recursive programming pioneered by LISP.

Now, importantly, it must be noted that the addition of sequence variables increases the expressive power of our framework beyond that of FOL. Notably, it is easy to show that the result of adding sequence variables to $\mathscr{L}_1$ with the above semantics makes the corresponding logic non-compact,[10] hence not first-order, and hence incapable of having a complete proof theory — a potentially serious blow to prospects of automated reasoning with respect to knowledge bases formulated with sequence variables. However, for the vast majority of practical purposes, as in the example (10) above, all that is needed are sentences in which all of the sequence variables are universally quantified with widest possible scope. And if the grammar of $\mathscr{L}_1$ + sequence variables is restricted accordingly, the resulting logic remains fully first-order.[11]

3.2 Adaptation II: Relaxed Typing — Predicates and Function Symbols

Further representational possibilities that can naturally arise on open networks bring the highly typed nature of traditional FOL into question as well. This was perhaps initially seen in the development of “frame-based” KR languages in which the traditional role of predicates in expressing properties was, in part at least, subsumed by what were essentially function symbols. In these systems, attributes are often thought of as functions on individuals (often called “slots” or “roles”) that yield “values” that, in a more traditional representation, would simply be a second argument to a predicate (see, e.g., the influential KL-ONE system of Brachman & Schmolze (1985)). This, in such a knowledge bases, 'Teacher' might indicate a *function* on individuals that returns, for a given argument, his or her (most influential, say) teacher, e.g.:

(16) Teacher(Aristotle) = Plato

instead of the more traditional sort of representation one might well find in another knowledge base:

(4) Teacher(Plato,Aristotle) .

Traditional FOL, with its strict typing, does not permit the predicate 'Teacher' simultaneously to play the role of a function symbol. Hence, to merge a knowledge base containing (4) with one containing (16) using a traditional first-order language would require considering the character string 'Teacher' in (4) to be a distinct lexical item from the identical string in (16). However, if one relaxes the strict syntactic typing of traditional FOL — if, that is, one allows certain lexical items to be members of more than one lexical category — there is no need to do so. One can simply allow 'Teacher' to play both roles, to be *both* a predicate and a function symbol, a member of *both* Cn and Fn.

More formally, let $\mathscr{L}_2$ be the result of modifying $\mathscr{L}_1$ by rejecting the strict syntactic typing of function symbols and predicates and allowing for the possibility that Fn and Rn have members in common and, hence, that one and the same symbol can play the syntactic roles of both function symbol and predicate.

Note that allowing this sort of "cross-categoricity" does not require any change in the definition of a interpretation (as modified to accommodate variable polyadicity in the previous section). Rather, as members of both the class of predicates and the class of function symbols, cross-categorical lexical items like 'Teacher' are simply assigned *two* extensions in an interpretation: a relation extension *qua* predicate and a function extension *qua* function symbol. One then evaluates an expression containing an occurrence of the item according to the role of that occurrence. (4), in particular, turns out true just in case, for any objects a and b in the domain of discourse, $\langle a,b\rangle \in$ $rext$('Teacher') and (16) just in case $fext$('Teacher')$(b) = a$ (truth conditions whose equivalence can be enforced if desired simply by requiring that, for cross-categorical lexical items $\varepsilon \in Fn \cap Pr$, $fext(\varepsilon) = rext(\varepsilon)$).

Of course, this is not a theoretically profound modification. And it may at first sight appear that there is little practical difference between, on the hand, allowing predicates and function symbols to be cross-categorical and, on the other, proscribing cross-categoricity and introducing distinct but (for all practical purposes) orthographically indistinguishable items to occur in different lexical categories. The former, however, is a more natural generalization of variable polyadicity: just as a single, variably polyadic lexical item can take any number of arguments rather than a distinct lexical item for each adicity, so a single, cross-categorical lexical item can play distinct syntactic roles rather than a distinct lexical item for each role. This therefore avoids any need for two different usages of the same symbol — or orthographically identical but theoretically distinct symbols — to be negotiated and separated. We elaborate on the importance of this point in the context of the Web in the following section, which introduces a much more profound relaxation of strict typing than cross-categoricity between function symbols and predicates.

3.3 Adaptation III: Complete Cross-categoricity and "Objectified" Relations

The breakdown of clear and inviolable boundaries between lexical types in the context of the Semantic Web extends well beyond function symbols and predicates to include terms as well. This turns out to have much deeper and more interesting syntactic and semantic implications.

In many systems, the traditional semantic values of predicates — properties, relations, classes of (*n*-tuples of) individuals — are treated as "first class citizens", i.e., as individuals. This is particularly common in Description Logics (see Baader et al. (2003)) which grew out of earlier knowledge representation systems that were originally designed to capture hierarchies of classes. Generally speaking, in these systems, quantifiers range over both individuals and classes and one is able to ascribe properties and relations to both. Thus, in such a system, one might well be able to say both that Plato is the teacher of Aristotle and that the Teacher relation is the converse of the Student relation. A knowledge base — one that might result, say, from merging a historical database with a more general ontology of education — might therefore very naturally contain both

(4) Teacher(Plato,Aristotle)

and

(17) ConverseOf(Teacher,Student)

Strict typing, of course, would prohibit this in a traditional first-order language. One way of dealing with this phenomenon, of course, is to move to a *second-order* language in which one distinguishes between *first-order* predicates that take individual constants and other first-order terms as arguments and *second-order* predicates that can take first-order predicates as arguments. There are, however, several problems with this. First, as is well-known, second-order validity is not axiomatizable, which means that one cannot make use of the wide array of first-order theorem provers that are available to support automated reasoning on information within a higher-order knowledge base. Second, just as one finds variation in adicity, a second-order framework would be forced to deal with variation in order as well. For instance, the predicates THING and ENTITY that are common in Web taxonomies and other knowledge bases are freely applied to items referring to individuals, classes, and relations alike. There is in general no practical way to attach a fixed order to a given predicate in a dynamic, anarchic environment like the Web. Best, therefore, if there were only one order to choose from. Hence, as far as possible, it is desirable (a) to remain first-order and, consequently, (b) to *loosen* traditional first-order syntactic and semantic restrictions as far as possible in response to representational pressures like the co-occurrence of sentences like (4) and (17).

Accordingly, in this spirit, just as the examples of the previous section led us to relax the partitioning of predicates and function symbols, let us accommodate examples like (4) and (17), not by moving to a strictly typed second-order language, but by *rejecting* the strict syntactic typing of traditional first-order languages and relaxing the divide between predicates and individual constants. More exactly put, to accommodate both (4) and (17), lexical items like 'Teacher' can be considered *completely* cross-categorical, that is, to be members, not only of the classes *Pr* and *Fn*, but of the class *Cn* of individual constants as well. Let $\mathscr{L}_3$ be the result of so modifying $\mathscr{L}_2$.

3.3.1 Complete Cross-Categoricity and the Web

As in the previous section, theoretically such phenomena could be accommodated by retaining strict typing and thinking of the occurrences of 'Teacher' in (4) and (17) as occurrences of distinct but orthographically identical expressions, one a predicate, the other an individual constant that might be related logically somehow. But this is simply not reflected on the Web.

A fundamental architectural assumptions of the Web concerns the notion of a Web *identifier*, referred to by the various acronyms 'URL', 'URN', and, more generally, 'URI' (see, e.g., W3C 1995/2005)). URIs are, essentially, names that, by means of rather baroque but well-defined syntactic conventions, encode enough information to make them globally univocal. The contast with traditional FOL, however, is rather striking. In traditional FOL, the choice of names has been viewed as essentially arbitrary and not a matter of the slightest logical or practical importance; indeed, in some well known theories — ZF set theory, for example — there are no names at all. But on the Web, names have an essential, and essentially public, role: their scope, once published, is the entire planet and the entire future. Indeed, names —URIs — are in a very real sense the fabric, the skeleton, of the Web. Whatever logic is implemented on the Web is merely infrastructure in support of this world-wide system of interrelated names.

The primary semantic assumption concerning URIs (and ensured in practice by the conventions just noted) is that every occurrence of a URI denotes the same entity in any context, regardless of the logical type of that occurrence. This is, of course, by design, as a mechanism guaranteeing that names consistently have the same semantic values regardless of context and syntactic type is essential for the sort of integration and interoperability across multiple knowledge bases that the Semantic Web is meant to foster. In this milieu, how cross-categorical phenomena are dealt with is no longer arbitrary or inconsequential. To the contrary, current Web theory and practice call for complete cross-categoricity.

3.4 Adaptation IV: Type-free Intensionality

The desire to express logical relations between relations, as in (17), is not the only way in which relations might be "objectified" — i.e., treated as individuals in their own right — in a knowledge base. The pervasive linguistic phenomenon of *nominalization* can generate examples of very natural examples of cross-categoricity (see, e.g., Chierchia & Turner (1988)). Consider:

(18) Whenever John is running, he hates it.

As this example shows, a gerund like 'running' can play both adjectival and nominal roles very naturally, even within the same sentence. Such examples are naturally formalized in a language allowing predicates to be cross-categorized with constants; (18), in particular, is naturally formalized as

(19) $\forall t(\mathrm{Time}(t) \supset (\mathrm{running}(\mathrm{John},t) \supset \mathrm{Hates}(\mathrm{John},\mathrm{running},t)))$.

Semantically, to accommodate examples like (4), (17), and (19) that can be expressed in $\mathscr{L}_3$, it would appear that no changes are required to the notion of interpretation developed for $\mathscr{L}_2$ (which, recall, simply involved modifying the original notion of an interpretation to accommodate variable polyadicity). Rather, members of $Pr \cap Cn$ simply have *both* a denotation (*qua* members of Cn) and a relational extension (*qua* members of Pr); members of $Fn_{\mathscr{L}} \cup Cn_{\mathscr{L}}$ have *both* a denotation and a function extension (*qua* members of Fn).

However, once completely cross-categorical lexical items are allowed that can play all three central grammatical roles in our Web-oriented language, a variety of intuitively valid cross-categorical inferences involving identity and quantification very naturally arise that that cannot be supported without ensuring a tighter connection between denotations and extensions. This requires a semantic adaptation paralleling syntactic cross-categoricity.

3.4.1 Identity and Type-freedom

Consider the following intuitively valid argument:

(20) Being married is the same as being hitched. Elvis and Priscilla are married. Therefore, Elvis and Priscilla are hitched.

Formalized, this argument would appear to be a straightforward instance of the logical principle of the intersubstitutivity of identicals, viz.,

(21) Married=Hitched, Married(Elvis,Priscilla) $\therefore$ Hitched(Elvis,Priscilla)

But the argument is not valid in the semantics of the previous section as it stands, as nothing about the relations expressed by 'Married' and 'Hitched' follows from the fact that their denotations are identical; that is, more specifically, from *den*('Married') = *den*('Hitched') it does not follow that *rext*('Married') = *rext*('Hitched'). Hence, it might in be the case that $\langle$*den*('Elvis'), *den*('Priscilla')$\rangle \in$ *rext*('Married') but $\langle$*den*('Elvis'), *den*('Priscilla')$\rangle \notin$ *rext*('Hitched'). Clearly, this won't do. Arguments of the above form involving property identities are natural and common, and are especially important for integrating diverse knowledge bases, where it is often the case that one and the same property is identified by different expressions.

The difficulty here is that the syntactic changes of the preceding section were not truly reflected in the semantics. To permit completely cross-categorical terms is to abandon the strict syntactic typing of traditional first-order languages— it is to say in particular that one and the same item can *simultaneously* be a constant and a predicate. By contrast, the semantics above preserved the traditional distinction between semantic types and simply allowed the semantic value of a cross-categorical term to be flatly ambiguous, taking on a different semantic value for each type. This gives rise to a disconnect that generates the problem above. The solution is a semantics that mirrors the syntax in rejecting the strict typing of traditional FOL and embraces *type-free* relations and allows that one and the same semantic entity can be *simultaneously* be — or perhaps more accurately, play the roles of — an individual and a relation. That is, for lexical items $\varepsilon \in Cn \cap Pr$ that can occur in both argument and predicate position in atomic formulas, it is stipulated that $den(\varepsilon) = rext(\varepsilon)$ — and hence that the relation $rext(\varepsilon)$ is a fully-fledged individual among others the domain D of **I**. From den('Married') = den('Hitched') it then follows immediately that $rext$('Married') = $rext$('Hitched') and, hence, that if ⟨den('Elvis'), den('Priscilla')⟩ ∈ $rext$('Married') then ⟨den('Elvis'), den('Priscilla')⟩ ∈ $rext$('Hitched').

But this move might raise some red flags. To objectify a property or relation, as the identity just noted implies, is to consider it to exist in the set D of individuals of an interpretation. Running, of course, is not the sort of thing that could itself ever be running. However, suppose, for example, there is a cross-categorical identity predicate in $\mathscr{L}$ and hence that there is an objectified identity relation $Id \in D$ that it denotes. As noted in §1, typically, in traditional FOL, properties and relations just *are* their extensions, i.e., sets of (n-tuples of) individuals in D. So understood, Id is simply the set $\{\langle a,a\rangle \mid a \in D\}$ of pairs of members of D. But since $Id \in D$, it follows that $\langle Id,Id\rangle \in Id$, violating the set theoretic axiom of foundation, which proscribes self-membership and other, more general forms of non-wellfoundedness. More generally, in the semantics of FOL, there are, typically, no restrictions on what individuals can occur in the extensions of what predicates. Hence, absent specific restrictions — which would seem entirely *ad hoc* — the denotation of any cross-categorical predicate can end up being a member of itself.

An obvious move here is simply to abandon the axiom of foundation and opt instead for conception of set that allows self-membership and other forms of non-wellfoundedness. Foreign as the suggestion might seem, the axiom of foundation is in fact entirely inessential to all applied mathematical purposes and, indeed, as the work of, notably, Aczel (1983) and Barwise & Moss (1996) have shown, set theories with *anti*-foundation axioms — i.e., axioms that posit the existence of non-wellfounded sets outright — turn out to have powerful applications in mathematics, theoretical computer science, and philosophy. But the philosophical assumptions underlying the Semantic Web's vision of distributed but integrated knowledge bases suggests a more satisfying solution.

3.4.2 Intensionality

A declarative knowledge base is (among other things) an attempt to characterize a set of concepts as richly as possible (or, at least, as richly as necessary for the purposes at hand), typically by means of axioms of some ilk. One of the promises of the Semantic Web is that one will be able to draw upon existing knowledge bases to flesh out one's desired set of concepts without having to do all the work from scratch. Intuitively, then, one begins with an informal set of concepts in mind and subsequently, and often incrementally, characterizes them with greater increasing rigor and precision. Typically, that is, the creators of a knowledge base are aware that their axioms are incomplete and often in need of further refinement.

On this understanding, concepts — functions and relations — are best understood *intensionally*, the extensionalism of traditional first-order semantics notwithstanding. A concept is initially identified and, in itself remains stable; it is only one's understanding of it — expressed in particular in the axioms one adds about it in a growing, dynamic knowledge base — changes over time. Additionally, even when fully characterized, a concept's extension can change over time simply in virtue of the dynamic nature of individuals — a person who was not a mother comes to have a child; a tree that had lacked leaves gains them; and so on. To accommodate this intuitive view of concepts — one that is ubiquitous in the Semantic Web community — the extensional conception of properties and relations is best abandoned: Functions, properties, and relations *have* extensions, but they are not themselves identical with those extensions.

The major technical modification is that function and relation extensions are now assigned to functions- and relations-in-intension directly, rather then to the function symbols and predicates that denote them. Thus, on this view, *Id* is an element of one of the ordered pairs in its own extension, but, as it is not identical with that extension, or with any set, it does not violate foundation.

More exactly, then, to accommodate intensionality in the notion of an interpretation **I**, separate sets F and R are introduced — the functions- and relations-in-intension of **I**, respectively — and the definition of the extension functions *fext* and *rext* is so altered that these functions apply to these new semantical entities directly rather than to function symbols and predicates:

- $fext : F \longrightarrow \{f \mid f : D^* \longrightarrow D\}$
- $rext : R \longrightarrow \{r \mid r \subseteq D^*\}$

Intuitively, as indicated in the preceding section, a cross-categorical function symbol that is also a predicate denotes a function that is also a relation. Accordingly, no restrictions are placed on F and R that would prevent a function from simultaneously being a relation, i.e., there is no restriction to the effect that $F \cap R$ is nonempty. Similarly, since objectified functions and relations occur in D, there can also be overlap between F and D and R and D.[12]

Recall that *fext* and *rext* had been the mechanisms for supplying meanings for function symbols and predicates. Accordingly, the denotation function *den* is extended so that it subsumes this functionality by defining it on all lexical items generally, appropriately restricted so as to assign an entity of the appropriate sort to each item. More exactly, that is, *den* is now defined to be a function on $Cn \cup Fn \cup Rn$ such that:

$den : Cn \longrightarrow D$

$den : Fn \longrightarrow F$

$den : Rn \longrightarrow R$

The definition of satisfaction for atomic formulas is now revised accordingly: For a given variable assignment s:

- s satisfies $\pi(\tau_1,\ldots,\tau_n)$ in $\mathbf{I}$ if and only if $s'(\tau_1)^\frown\ldots^\frown s'(\tau_n) \in rext(den(\pi))$.

3.4.3 Logical Concerns

In fact, an intensional conception of concepts not only comports better with KR practice, it is arguably theoretically preferable (or at least, more familiar) as well, the idea can be implemented semantically with no violation of the axiom of foundation. *Id*, in particular, viewed as a relation-in-intension rather than a set of ordered pairs, is no longer a member of a member of itself but rather simply a member of an ordered pair — viz., $\langle Id,Id\rangle$ — in its *extension*. As such, intuitively, *Id* bears the relation of identity — i.e., itself — to itself, but that is simply an instance of *self-exemplification* and involves no violation of any set theoretic principles. It might, however, raise a couple of logical concerns.

The first has to do with cardinality. Given the principle that every subset of D^* is the extension of at least one relation, it follows that, for an interpretation **I** with domain D:

(22) There are as many relations as there are subsets of D^*.

If so, however, there are more relations than individuals. For by simple transfinite arithmetic, if m is the cardinality of D (recall that, by definition of an interpretation, D must be nonempty, so $m > 0$), the cardinality of D^* is at least m and, hence, by Cantor's famous theorem (which still holds in non-wellfounded set theories), there are more than m subsets of D^*. Hence, by, (22), there are more relations than there are elements of D. Hence, on pain of paradox, it is not possible to think of relations as individuals in the manner suggested.

The argument is specious. There are, of course, more *extensional* relations over D than there are elements of D. However, first, there is no reason to accept the principle that, for every arbitrary set of individuals there is some *intensional* property they uniquely share in common. But even if this principle is true in some deep, metaphysical sense, nothing about the semantics above requires that *all* properties and relations be in D. To the contrary, in order to represent any piece of information that one is likely to want to include in a given knowledge base, there is no reason to assume that there are any more relations than those that can be *named* in one's language $\mathscr{L}$. That is, one needn't assume there are any more relations than there are predicates of $\mathscr{L}$. This leaves us free instead to assume, without any obvious cardinality problems, that among the objects in D are as many relations as there are cross-categorical lexical items of $\mathscr{L}$.

That one needn't assume the existence of anything other than explicitly named relations is also what alleviates concerns over logical paradoxes like the intensional version of Russell's paradox: given the property R of *nonselfexmplification*, it is easy to show that R both does and does not exemplify itself. However, Russell's paradox, as well as all other logical paradoxes, can only arise in a formal framework in which it is possible to name or otherwise prove the existence of R or some similarly problematic entity.[13] Lacking any such capacity, no threat of paradox looms for $\mathscr{L}$ and its semantics.

3.4.4 “Higher-order” Quantification and Its Logical Consequences

It is a prominent feature of FOL that it supports existential generalization on argument places. This implies that the nominal occurrence of ‘running’ in (19) can be existentially generalized, yielding:

(23) $\exists x[\forall t(\mathrm{Time}(t) \supset (\mathrm{running}(\mathrm{John},t) \supset \mathrm{Hates}(\mathrm{John},x,t)))]$

that is, in English,

(24) There is something that John hates whenever he is running.

(24) is of course intuitively entailed by (18) and, fittingly, (23) is validly entailed by (19) on the semantics of the previous section. However, intuitively, (18) entails something stronger, namely, that what John hates when running is *exactly that*, viz., running. That is, intuitively, as with (21), the two occurrences of ‘running’, despite playing different syntactic roles, have the *same* semantic value. That is, it is possible to infer not only (24) from (18) but also something like:

(25) There is something that John hates whenever he is doing it,

where *both* occurrences of ‘running’ in (18) are generalized upon. Accordingly, it should be possible formally to infer a corresponding representation of (25) from (19) in which on both occurrences of ‘running’ are existentially generalized, e.g.:

(26) $\exists F[\forall t(\mathrm{Time}(t) \supset (F(\mathrm{John},t) \supset \mathrm{Hates}(\mathrm{John},F,t)))]$.

But this is currently not possible in our language as it stands, as variables are only allowed to occur in argument position.

Again, a knowledge base containing both (4) and (17) entails

(27) $\mathrm{ConverseOf}(\mathrm{Teacher},\mathrm{Student}) \wedge \mathrm{Teacher}(\mathrm{Plato},\mathrm{Aristotle})$.

Hence, intuitively, Plato and Aristotle stand in a relation whose converse is the Student relation. Thus, as with the move from (19) to (26), it should be possible to generalize on both the nominal and predicative occurrences of ‘Teacher’ in (27):

(28) $\exists F(\mathrm{ConverseOf}(F,\mathrm{Student}) \wedge F(\mathrm{Plato},\mathrm{Aristotle})$.

Relatedly, it should be possible to infer from (27) that

(29) Student(Aristotle,Plato) .

And this of course follows directly if the converse relation is axiomatized in the obvious way:

(30) $\forall F \forall G(\text{ConverseOf}(F,G) \supset \forall x \forall y(Fxy \supset Gyx))$

All three of these examples can be accommodated syntactically in our type-free framework by means of a simple adaptation that allows variables themselves to be cross-categorical, that is, to occur, not only in argument position, but also in function and predicate position as well. With this adaptation, (26), (28), and (30) are all syntactically legal.

A concomitant semantic adaptation ensures that the three inferences above are all valid. With the relaxation of traditional typing distinctions to allow both type-free lexical items and corresponding, type-free properties and relations, these cases require only two simple semantic tweaks. First, for a variable assignment *s*, the extension *s*′ is defined so that it applies not only to terms but to function symbols and predicates as well.[14] The satisfaction clause for atomic formulas is then restated accordingly:

- *s* satisfies $\rho(\tau_1,\ldots,\tau_n)$ in **I** if and only if $s'(\tau_1)^\frown\ldots^\frown s'(\tau_n) \in s'(\rho)$.

With this modification, (26) can be shown to follow straightaway from (19), (28) from (27), and (29) from (27) and (30).

4 Common Logic

As can now be seen, a mix of cross-categorical and non-cross-categorical lexical items in a first-order language requires a rather finicky semantics. However, having recognized the need for cross-categoricity, it is arguable that a general logical framework for the Web should by default be fully type-free, that is, by default, there should be no syntactic and semantic typing whatsoever. This is the standpoint of Common Logic (ISO 2007), a recent ISO standard for publishing and exchanging logic-based information on the Web that embodies all of the adaptations discussed above.[15]

4.1 Complete Type Freedom

There are two reasons for full type-freedom in a logical framework for the Semantic Web. First, while authors can be expected to comply reasonably well with web-based syntactic standards if flouting them prevents them from achieving their desired ends —

as can occur, for example, if one attempts to publish an HTML document lacking a proper header — one cannot realistically expect a logical knowledge base to comply with even long-standing and well known syntactic standards in logic, especially knowledge bases that might have multiple authors and that might have developed gradually over time. And indeed it is not uncommon to find knowledge bases in which any given lexical item is used indiscriminately to play two or even all three lexical roles traditionally divided among predicates, function symbols, and individual constants.

Secondly, however, even if adherence to traditional syntactic structures could be enforced *within* knowledge bases, in an open network, there is simply no way to ensure that the same term is used only as a constant, predicate, or function symbol *across* knowledge bases. Hence, if knowledge bases are to be integrated — as is a major goal of the Semantic Web — the possibility of cross-categoricity must be allowed. But, because it is impossible in general to anticipate which lexical items might end up being cross-categorical, and because, in principle, any (nonlogical) item could, it makes sense simply to take *all* nonlogical lexical items to be cross-categorical. Complete type-freedom, that is, should be the default and typing restrictions should simply be tacked on as needed as the special case.

The corresponding semantic thesis is that there is no longer any distinction made between individuals, functions, and relations. There are simply objects, first-class citizens: Just as any name can play any of the three major syntactic roles — argument, function symbol, predicate — so any object can play any of the three major semantic roles — individual, function, relation.

4.2 A Fully Type-free Logic

To capture the idea, the lexicon of a language is now defined to consist of nothing other than a single nonlogical lexical category N of *names*. Even the distinction between names and variables can go by the wayside, for on the Web, absent any sort of shared convention, there is no way to distinguish names from free variables. The simplest way of dealing with this is simply for names to subsume the usual role of variable: like variables, they can all be quantified. Unquantified, names function as usual.[16] As there is no finite bound on the number of quantifiers that can occur in a sentence, it must be assumed that $N_{\mathscr{L}}$ is countably infinite.

4.2.1 CL Grammar: A Family of Dialects

A final, very important feature of CL is also motivated by the rise of KR languages their use to represent information on the Web. The rise of many distinct KR languages with many different features led to many internecine conflicts over whose language was superior to whose. While these debates were sometimes based in matters of genuine substance, often as not disagreements were rooted in mere matters of form, such as the superiority of LISP-like languages over traditional languages in the *Principia Mathematica* style. While there may be practical advantages of one language over another in various contexts, at root KR languages are theoretically similar, if not identical. Rather than spawn yet another series of debates over form, early in the development of CL it was decided to express the syntax of CL languages (or *dialects*, in CL's terminology) in purely abstract terms so as to encompass a boundless variety of concrete representation languages, from LISP-like languages to graphical languages and everything in between.

Specifically, then, let *N* be nonlogical lexicon. The members of *N* are called *names*, one of which, referred to as *Id*, is distinguished. The logical lexicon of every CL dialect will include a countable set of sequence variables, or sequence *markers* in CL's terminology, and five *connective types* — conjunction, disjunction, implication, biconditional, and negation — and two *quantifier types* — existential and universal. A CL *dialect* (*based on N*) must meet the following conditions:[17]

1. Every name is a *term* (*of* $\mathscr{L}$).

2. A *term sequence* is a finite sequence of terms or sequence markers.

3. A *functional term* consists of a term, called the *operator* (of the term), and a term sequence, called the *argument sequence*, the elements of which are called the *arguments*.

4. A *sentence* is either an atomic sentence, a Boolean sentence, or a quantified sentence.

5. An *atomic sentence* consists of a term, called the *predicate* (of the sentence), and a term sequence, called the *argument sequence*, the elements of which are called the *arguments*.

6. A *Boolean sentence* has a unique connective type and a number of sentences, called the *components* (of the sentence). The number of components depends on

the type: conjunctions (i.e., sentences whose connective type is conjunction) and disjunctions can have any number; implications and biconditionals must have exactly two; negations must have exactly one.

7. A *quantified sentence* has (i) a unique quantifier type, (ii) a finite, nonrepeating sequence of names and sequence markers, called the *binding* sequence (of the sentence) and (iii) a sentence, called the *body*.

Variable bondage and freedom are understood as usual. As in §3.4.2, the extension functions *fext* and *rext* are taken to be functions on semantic objects directly, rather on lexical items. However, in the spirit of full type-freedom, there is no longer any distinction made between individuals, functions, and relations. There are simply objects, first-class citizens, each of which is assigned a function extension and a relation extension. Semantically, therefore, an interpretation **I** for $\mathscr{L}$ it taken to be a 4-tuple $\langle D, den, fext, rext\rangle$, where *den*, more or less as before, is a function on the names of $\mathscr{L}$ and *fext* and *rext* are taken to be functions on all of D:

- $den : N_{\mathscr{L}} \longrightarrow D$.
- $fext : D \longrightarrow \{f \mid f : D^* \longrightarrow D\}$
- $rext : D \longrightarrow \{r \mid r \subseteq D^*\}$

Truth for the sentences of a CL dialect $\mathscr{L}$ in an interpretation **I** is defined in a natural way. Since there is no distinct class of variables, the concept of a variable assignment goes by the board.[18] Instead, the definition of a ν-variant is altered and generalized so that it applies to interpretations directly. Specifically, for a sequence $\langle \nu_1,\ldots,\nu_n\rangle$ of names or sequence markers, say that $\mathbf{I}^* = \langle D, den^*, fext, rext\rangle$ is a $\langle \nu_1,\ldots,\nu_n\rangle$*-variant* of **I** just in case $den^*(\mu) = den(\mu)$ for every name or sequence marker μ distinct from the ν_i, i.e., just in case den^* differs from *den* at most in what it assigned to the names and sequence markers of $\langle \nu_1,\ldots,\nu_n\rangle$. Then denotation for terms generally and truth for sentences is defined as follows.

1. If $\varepsilon \in N_{\mathscr{L}}$, then $d(\varepsilon) = den(\varepsilon)$.

2. If τ is a functional term and α its function term and $\langle \tau_1,\ldots,\tau_n\rangle$ its argument sequence, then $d(\tau) = d(\alpha)(d(\tau_1),\ldots, d(\tau_n))$.

3. If φ is an atomic sentence and π its predicate and $\langle\tau_1,\ldots,\tau_n\rangle$ its argument sequence, then φ is *true in* **I** if and only if $\langle d(\tau_1),\ldots, d(\tau_n)\rangle \in \mathit{rext}(d)(\pi)$.

4. If φ is a conjunction, then φ is true in **I** if and only all of its components are. Similarly for the remaining Boolean types.

5. If σ is quantified sentence whose type is universal (existential) whose binding sequence is $\langle\nu_1,\ldots,\nu_n\rangle$ and whose body is ψ, then φ is true in **I** if and only if ψ is true in every (some) $\langle\nu_1,\ldots,\nu_n\rangle$-variant **I*** of **I**.

Despite their syntactic freedom and apparently higher-order character, CL dialects without sequence variables (or with the syntactic restriction discussed at the end of §3.1.1) are entirely first-order and hence complete. [19]

5 A Brief Final Judgment

In this paper, I have traced a series of evolutionary adaptations of FOL motivated entirely by its use by knowledge engineers to represent and share information on the Web culminating in the development of Common Logic. While the primary goal in this paper has been to document this evolution, it is arguable, I think that CL's syntactic and semantic egalitarianism better realizes the goal "topic neutrality" that a logic should ideally exemplify — understood, at least in part, as the idea that logic should as far as possible not itself embody any metaphysical presuppositions. Instead of retaining the traditional metaphysical divisions of FOL that reflect its Fregean origins, CL begins as it were with a single, metaphysically homogeneous domain in which, potentially, anything can play the traditional roles of object, property, relation, and function. Note that the effect of this is not to *destroy* traditional metaphysical divisions. Rather, it simply to refrain from building those divisions explicitly into one's *logic*; instead, such divisions are left to the user to introduce and enforce axiomatically in an explicit metaphysical theory.

CL's relevance thus arguably extends beyond the knowledge engineering community to philosophy proper, as it gives philosophers a more flexible logical framework for the representation of philosophical content as well.

References

Aczel, P. (1983). *Non-well-founded sets*. (Stanford, CA: CSLI Publications)

Baader, F., D. Calvanese, D. McGuinness, D. Narde, and P. Patel Schneider (Eds.) (2003). *The description logic handbook: Theory, implementation and applications*. (Cambridge: Cambridge University Press)

Barwise, J. & Moss, L. (1996) *Vicious circles: On the mathematics of non-wellfounded phenomena*. (Stanford, CA: CSLI Publications)

Bealer, G. (1982). *Quality and concept*. (Oxford: Oxford University Press)

Brachman, R. & Schmolze, J. (1985). An overview of the KL-ONE knowledge representation system." *Cognitive Science*, 9, 171–216.

Chierchia G. & Turner, R. (1988). Semantics and property theory. *Linguistics and Philosophy*, 11, 261–302.

Genesereth, M. (1998). Knowledge interchange format. Draft proposed American National Standard (dpANS), NCITS.T2/98-004. Retrieved January 9, 2009 from http://logic.stanford.edu/kif/dpans.html.

Grandy, R. E. (1976). Anadic logic and English. *Synthese* 32, 395–402.

Gruber, T. R. (1992). Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Stanford University, Knowledge Systems Laboratory.

Hayes, P. (2003). RDF semantics. W3C Technical Report, Retrieved January 15, 2009 from the W3C web site: http://www.w3.org/TR/rdf-mt.

ISO (2007). Information technology — Common Logic (CL): A framework for a family of logic-based languages." International Standard ISO/IEC 24707, First edition, 2007-10-01, Retrieved January 9, 2009 from the ISO web site: http://standards.iso.org/ittf/PubliclyAvailableStandards/c039175_ISO_IEC_24707_2007(E).zip.

Kenny, A. (1963). *Action, emotion and will*. (London: Routledge and Kegan Paul)

Lenat, D. & Guha, R. V. (1991). The evolution of CycL, the Cyc representation language. ACM SIGART Bulletin 2, 84-87.

Lindström, P. (1969). On extensions of elementary logic. *Theoria* 35, 1-11.

MacGregor, R. & Bates, R. (1987). The LOOM knowledge representation language. Technical Report ISI/RS-87-188, USC/Information Sciences Institute.

Menzel, C. (1993). The proper treatment of predication in fine-grained intensional logic. (In J. E. Tomberlin (Ed.), *Philosophical perspectives*, volume 7 (pp. 61–87). Atascadero: Ridgeview Publishing Company.)

Menzel, C. & Hayes P. (2003). SCL: A logic standard for semantic integration". (In A. Doan, A. Halevey, and N. Noy (Eds.), *CEUR Workshop Proceedings: Semantic Integration*, volume 82. Available online at http://ceur-ws.org/Vol-82.)

Patel-Scheider, P., McGuinness, D., Brachman, R., & Resnick L. (1991). The CLASSIC knowledge representation system: guiding principles and implementation rationale. *ACM SIGART Bulletin* 2, 108-113.

Shapiro, S. C. (2000). SNePS: A logic for natural language understanding and commonsense reasoning. (in L. M. Iwanska & S. C. Shapiro (Eds.), *Natural language processing and knowledge representation: Language for knowledge and knowledge for language* (pp. 175-195). Menlo Park, CA/Cambridge, MA: AAAI Press/MIT Press.)

Sowa, J. F. (1984). *Conceptual structures: Information processing in mind and machine*. (Reading, MA: Addison-Wesley)

W3C (1995). RFC: 1808: Relative uniform resource locators (RFC 1808). Retrieved January 15, 2009 from the W3C web site: http://www.w3.org/Addressing/rfc1808.txt

W3C (2005). RFC 3986: Uniform resource identifier (URI): Generic syntax. Retrieved January 15, 2009 from the W3C web site: http://labs.apache.org/webarch/uri/rfc/rfc3986.html

Notes

[1] Some approaches in fact treat constants (rather naturally) as 0-adic function symbols.

[2] Viewed as sets of *n*-tuples, of course, functions can be seen as a type of relation, so the syntactic division of function symbols and predicates may not be quite as absolute in the semantic domain.

[3] More exactly, the "functionality" condition that a 1-place function — conceived as a set of ordered pairs — must meet is that, if $\langle a,b \rangle$ and $\langle a,c \rangle$ are members of f, then we must have $b = c$.

[4] Where $\mathbb{N}^+$ is the set of positive integers and $D^1 = D$, $D^2 = D \times D$, $D^3 = D \times D \times D$, etc.

[5] Specifically, for constants κ, $s'(\kappa) = den(\kappa)$, for variables ν, $s'(\nu) = s(\nu)$, and for function terms $\alpha(\tau_1,\ldots,\tau_n)$, $s'(\ulcorner\alpha(\tau_1,\ldots,\tau_n)\urcorner) = fext(\alpha)(s'(\tau_1),\ldots, s'(\tau_n))$. We are obviously being careless about quotation here.

[6] Specifically, if b and c are distinct individuals, then an extensional function f cannot contain both $\langle a_1,\ldots,a_n,b\rangle$ and $\langle a_1,\ldots,a_n,c\rangle$, for all $n \geq 1$.

[7] Grandy's anadic logic is actually a variable-free algebraic logic, but it is easily converted into an equivalent but more traditional first-order logic.

[8] In KIF sequence variables were restricted to occur only in the last argument position in any atomic sentence; this restriction has been relaxed in CL.

[9] Note that, where $\tau_1, \ldots, \tau_n$ are all ordinary individual terms, $s'(\tau_1)^\frown\ldots^\frown s'(\tau_n)$ is just $\langle s'(\tau_1), \ldots, s'(\tau_n)\rangle$, so the clause for atomic formulas here is a simply generalization of the original clause.

[10] A logic is *compact* if an arbitrary set of sentences of the logic has a model if all of its finite subsets do. By a famous theorem of Lindström (1969), a logic is first-order only if it is compact. To see that a logic with sequence variables (absent any grammatical restrictions – see below) and the above semantics for them is not compact, let S be the set

$$\{\exists s P(s), \forall x_1{\sim}P(x_1), \forall x_1x_2{\sim}P(x_1,x_2), \forall x_1x_2x_3{\sim}P(x_1,x_2,x_3), \ldots\}.$$

Clearly, every finite subset S* of S has a model but S itself does not, as its universally quantified sentences jointly entail that no finite sequence of individuals is in the relational extension of 'P', i.e., they jointly entail that '∃sP(s)' is false.

[11] Let $\mathscr{L}_1^+$ be $\mathscr{L}_1$ + sequence variables where its grammar is so restricted. To see in particular that the logic for $\mathscr{L}_1^+$ is compact, note simply that every sentence φ of $\mathscr{L}_1^+$ is logically equivalent to a set of sentences obtained by explicitly unpacking the quantified sequence variables of φ. (For example, the sentence '∀sP(s)' will be true in an interpretation **I** if and only if **I** is a model of the set $\{P(), \forall x_1 P(x_1), \forall x_1x_2 P(x_1,x_2), \ldots\}$.) Since the logic for $\mathscr{L}_1$ is compact, every set of sequence-variable-free sentences of $\mathscr{L}_1^+$ has a model if all of its finite subsets do. Hence, the same is true for any set S of sentences of $\mathscr{L}_1^+$, since we can unpack all of the sentences of S that contain sequence variables into sequence-variable-free sentences.

[12] See Bealer (1982) and Menzel (1993) for more philosophical motivations for a type-free, intensional semantics.

[13] Paradoxes in set theory, in particular, arise from overly promiscuous set existence principles like the so-called naïve comprehension schema which generates a set $\{x : \varphi\}$for every expressible condition φ. From this principle the existence of the Russell set of all nonselfmembered sets follows immediately from the condition $x \notin x$.

[14] More exactly: For $\kappa \in Cn$, $s'(\kappa) = den(\kappa)$, for $\alpha \in Fn$, $s'(\alpha) = fext(\alpha)$, for $\pi \in Pr$, $s'(\pi) = rext(\pi)$, and for variables ν, $s'(\nu) = s(\nu)$. Then, recursively, for function terms $\ulcorner\beta(\tau_1,\ldots,\tau_n)\urcorner$, $s'(\ulcorner\beta(\tau_1,\ldots,\tau_n)\urcorner) = s'(\beta)(s'(\tau_1)^\frown\ldots^\frown s'(\tau_n))$.

[15] Common Logic is a product of a fairly large working group that evolved from two projects to develop parallel ANSI standards for conceptual graphs (Sowa 1984) and KIF (Genesereth 1998). Eventually, those projects were merged into a single ISO project. Hayes and Menzel (2001) defined a very general model theory for a precursor to CL which Hayes (2003) used to define the semantics for the W3C languages RDF(S) and OWL. In addition to defining the general syntax and semantics for all CL dialects, the CL standard specifies three concrete dialects that are capable of expressing the full CL semantics: the Common Logic Interchange Format (CLIF), the Conceptual Graph Interchange Format (CGIF), and the XML-based notation for CL (XCL).

[16] Note that, semantically, this will be equivalent to one of two standard approaches to free variables. On one treatment, free variables are implicitly universally quantified; on the other, free variables are, in effect, names, and the combination of an interpretation plus a variable assignment essentially yields the approach here.

[17] In fact, CL dialects are defined more flexibly than this and encompass traditional, typed first-order dialects with fixed arities as well as type-free, arity-free dialects like CLIF. For purposes here, where the emphasis is on non-traditional adaptations (and for the sake of space), the less general definition above is used.

[18] Variable assignments exist for largely contingent historical reasons, rather than theoretical ones, and are frankly unnecessary in traditional FOL.

[19] The easiest way to demonstrate this is simply to translate a CL dialect into a traditional first-order language as follows. Let $\mathscr{L}$ be a CL dialect. Let $\mathscr{L}^*$ be a traditional first-order language in which all the names of $\mathscr{L}$ are individual constants and which contains countably many predicates Holds_1, Holds_2, … and countably many predicates App_1, App_2, …. Then define a translation function * from $\mathscr{L}$ to $\mathscr{L}^*$ as follows: For each name κ of $\mathscr{L}$, let $\kappa^* = \kappa$; for every complex term $\alpha(\tau_1,\ldots,\tau_n)$ of $\mathscr{L}$ let $\alpha(\tau_1,\ldots,\tau_n)^* = \mathsf{App}_n(\alpha^*,\tau_1^*,\ldots,\tau_n^*)$; and for every atomic formula $\pi(\tau_1,\ldots,\tau_n)$ of $\mathscr{L}$, let $\pi(\tau_1,\ldots,\tau_n)^* = \mathsf{Holds}_n(\pi^*,\tau_1^*,\ldots,\tau_n^*)$. Boolean and quantified sentences are translated recursively in the obvious way. It is easy then to define a method for translating any interpretation **I** of $\mathscr{L}$ into a unique interpretation **I*** of $\mathscr{L}^*$ and show that a sentence φ of $\mathscr{L}$ is true in **I** iff φ^* is true under **I***.