Cite as: Bergman, M. K. Potential Uses in Depth. in *A Knowledge Representation Practionary: Guidelines Based on Charles Sanders Peirce* (ed. Bergman, M. K.) 343–369 (Springer International Publishing, 2018). doi:10.1007/978-3-319-98092-8_16

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

Full-text: http://www.mkbergman.com/publications/akrp/chapter-16.pdf

Abstract: The three areas covered in depth in this chapter are workflows and business process management , semantic parsing, and robotics. The production and consumption of knowledge should warrant as much attention as do the actions or processes on the factory floor. We can map the compositional and semantic aspects of our language to the categorial perspectives of Peirce's logic and semiosis, and then convert those formalisms to distributions over broad examples provided by KBpedia's knowledge. Cognitive robots embrace the ideas of learning and planning and interacting with a dynamic world.

Robots, to conduct actions, must resolve or 'ground' their reasoning symbols into indecomposable primitives. The implication is that higher order concepts are derivations in some manner of lower level concepts, which fits KBpedia and Peirce's universal categories. Kinesthetic robots may also be essential to refine natural language understanding.

16

POTENTIAL USES IN DEPTH

Information, Peirce tells us, is like a spatial function covering the complete *area* of a topic. We have just covered a dozen topics in *breadth* regarding the potential of Peircean ideas to the subjects of knowledge and its representation. Now, let's turn our attention to the treatment of three additional topics, only now in *depth*. These three new topics are not as speculative as some from the prior chapter. I have chosen them based on highest impact, near-term potentials. The three areas are workflows and business process management (BPM), semantic parsing, and applications and interactions in robotics. The challenge in all three areas is to automate as much as possible while leaving to the knowledge worker what is uniquely human. For space reasons, I shorten the two bookends to allow a fuller treatment of the middle case.

WORKFLOWS AND BPM

Business processes and their management is the most neglected area of business. Business process management, or BPM, is about how businesses identify, select, implement, and refine their bases of production, and the actions to perform them. BPM embraces two kinds of knowledge: that needed to do a task, and the knowledge of what to do when a process goes off track. BPM is the *action* side of the business, analogous to the *things* of business, which is what KM manages.* Indeed, in this very manner of *verb-noun,* we see similar failings and lack of attention for BPM as we see for knowledge management.† It is telling that service-oriented, knowledge-based businesses still do not see that the fundamental basis of their products is knowledge. Attending to the production and consumption of knowledge warrants as much attention to efficiencies as do the actions or processes on the factory floor.

We first touched upon the subject of content workflows in *Chapter 12*. Here, I give

* BPM may also refer to business process *modeling*. We retain the management sense here, noting the modeling part only comes after thinking through the management portion.

† The exception to this observation is advanced manufacturing. Some of these businesses, inherently action-oriented, have pioneered BPM's related cousin of manufacturing process management. However, it is an open question whether manufacturing businesses are better at KM as well.

better detail to these flows and argue that knowledge management itself deserves business process management. Ideas in Peirce and KBpedia enable us to better represent actions and events, critical aspects of workflow. We will see real differences from BPM once we learn how to incorporate it into our actual, daily workflows.

We may group BPM activities in many ways but, fundamentally, they represent how to satisfice multiple business objectives, not always in concert. The goals of profit and public service, for example, are not weighed equally by companies and non-profit institutions. A business process initiative should consider a scope that at least includes:

1. A logical conceptual model of process and terminology for the business process agreed by the user community of workers and managers;
2. An agreed design and implementation plan;
3. Technology support to implement that system;
4. How users and administrators interact with the system, including user interfaces and approval steps and actions; and
5. Agreed and documented process and governance.

Initiatives require both semantic technologies and management commitment.

Today, however, we are not even at the preliminaries. For the majority of companies, agreed workflow procedures for business processes or operational workflow management systems do not exist. Gaps arise because BPM deals with abstract processes and intimately involves people, work practices, and management. Workflows cut across organizational boundaries and thus need to be attentive to terminology and semantics. To raise the question of 'what is your workflow?' is to disrupt the workflow of your standard knowledge worker. Many BPM efforts are bass-ackwards. We do not need a separate application upon which to focus our 'workflow' attention. We need workflow considerations rooted in how we currently work. While it is OK to disrupt the knowledge worker for a short period to help understand their implicit workflows, it is not OK to put in place BPM systems that divert knowledge workers from their standard work.

Semantic technologies are essential to the task because shared communication is at the heart of workflow management. Semantic business process management has been a steady research interest over the past 15 years or so (see Heppe *et al.* for one of the seminal early papers[1]). One theme is the potential role of ontologies in BPM.[2] Through this work we learn the imperative of incorporating 'action' in our ontologies and the need to handle branching and merging in evolving workflows. Unfortunately, however, besides some notable research initiatives from Europe, we have not seen broad commercial success for semantic BPM.

Concepts and Definitions

We focus on digital *content* in the context of BPM for knowledge work, which refers to all documents, datasets, records within them, the ontologies used by the system, and internal control vocabularies and structures. A *content lifecycle* embraces all content stages and workflow steps within those stages, from inception to use. In its entirety, we capture a content lifecycle by the complete *controlled vocabulary* and sequencing of all stages and steps. A *content stage* is a broader concept than workflow step and represents a state within the content lifecycle ranging from experimental to working to archived. *Figure 12-2* is one example of various content stages.

A *state* is an instantiation of a workflow step that represents a change in content or an evaluation of it. (For example, a manager approving the release of content even without making any changes to the actual content.) An *event* is an occurrence that causes a change in state. A *workflow* is a sequence of connected steps representing a business activity where each step follows without delay or gap and ends just before the next subsequent step may begin. A *workflow step* is a discrete step in a workflow that has an explicit label, where the general progression proceeds from creation through editing and review to approvals. Workflow step is a narrower concept than content stage within a content lifecycle. We organize workflows around discrete and definable business objectives such as authoring, harvesting (ingest), archiving, and the like. Upon completion of various workflows, we may deem the content ready for different stages in a content lifecycle. In an authoring workflow, for example, new content may proceed from creation to editing to completion of tagging and then review (with potential approval). Approval of authored content thus represents a change in the content stage from working to readiness for public release and use.

We can thus see the 'stages' of a content lifecycle as a broader organizing framework than the individual 'steps' of the workflows we embed within these content stages. All workflows, though individual steps may differ, share the basic conceptual progression of drafting or creation proceeding through editing or modification and then to testing and review before acceptance for public use or readiness. Because of these conceptual similarities, we can develop and share controlled vocabularies to represent these progressions across workflows, preferably using consistent terminology we draw from actual work practice.

We need to intimately relate these stages and steps to governance and quality control. We introduce known checks, reviews, and sign-offs into the workflow to ensure releases meet the organization's quality standards. 'Business processes' are the combination of an orderly progression in workflows with governance. By adopting such BPM practices, we help ensure the repeatability and generalizability of our organizational efforts.

Providers have developed *workflow engines* to keep these specifications persistent and to execute some of them, incorporating the various decision rules and triggers that enable the progression to proceed step-by-step through the workflows. The workflow engine is a software application that manages and executes modeled computer processes, and thus provides a coherent and standard way for specifying various business workflows and then executing them depending on the governance and quality checks desired by the organization. We may base events or triggers for moving from state to state within the progression on user or manual review, timing or duration checks, applications of scripts or automated tests, and the like. *Workflow management* is the collection of processes and governance by which we oversee the entire content lifecycle, including guidance for how steps progress across the cycle and how we conduct reviews and approvals.

We may represent these steps and activities with controlled vocabularies or graphic notations. *BPEL* (business process execution language) was the leading standard in the early days of service-oriented architectures. Today, *BPMN* (business process model and notation) is the leading approach whereby we use graphical editors to create visual workflows that generate XML instructions to the workflow engine. Optionally, we may represent the individual steps in a workflow process using a *work breakdown structure* (WBS), a management approach used for many decades.

The BPM Process

The business process management (BPM) cycle should begin with a *logical model*, starting, in the case of KM, from the viewpoint of content lifecycle and work stages. We should involve creators and users of the content, especially including responsible managers, in agreeing to the major stages and terminology of this model, as well as transition and decision points for state changes. We need to inspect and define specific stages of this cycle; adding steps to the process requires consultation with the stakeholders. The products of this part of the cycle are the initial controlled vocabularies and representations of content stages, similar to what we show in *Figure 12-2*. The result of this process creates the 'backbone' to the overall BPM effort.

We then need to express this logical model and its controlled vocabulary and terminology in an ontology to take advantage of semantic technologies. The stages and steps naturally lend themselves to class specifications. Review and approval levels become properties, all again governed by the agreed common vocabularies. At this juncture, we advise to discuss and decide upon required or optional annotation guidelines and metadata for events and state transitions. Noting the name of the approving employee and timestamps are a couple of standard fields. Rejection or re-review steps may warrant more elaborate notes. Some of these annotation standards may require input from legal counsel or be attentive to the regulatory requirements of the business.

It is important to balance current terminology in use with consistency across the full business process for:

- Content lifecycle stages;
- Workflow steps;
- Events;
- Actors (agents) and roles; and
- Transitions and alerts.

We must update the logical model and ontologies to reflect any changes to the conceptual model. The net result should be an updated workflow model in specification form that uses consistent terminology. This model will now become the baseline for workflows moving forward, which we may further refine or update or use as templates for other workflows as needed.

We configure the workflow engine in parallel to this effort. The major aim is to add the workflow logic and business rules, in addition to the install and configuration parameters. Many proprietary and open source workflow engines exist for both BPEL and BPMN, though choosing the right one is a complicated task involving a panoply of anticipated functions and the types of integration desired with other tools (such as graphical changes to a BPMN specification flowing through to the engine). As noted earlier, we have seen semantic integrations in research projects, but we appear to lack turnkey off-the-shelf systems that incorporate semantics. Aside from the need to integrate a workflow engine, prior chapters provide the guidelines for one.

Optimal Approaches and Outcomes

One has to wonder about the relatively low uptake of BPM for knowledge management functions. I think we can point to two reasons for this lack. First, as we discussed for the lack of use of information and knowledge in *Chapter 3*, managers do not have a bred-in-the-bones belief in the importance of process and workflows in knowledge management. Somehow we know we are involved with important tasks of discovery and pursuit of knowledge, but we do not understand these are purposeful and refinable activities. Second, I think the implementation of BPM has been bass-ackwards. Business process or workflows are not applications; they are an articulation of what we do. Recording or changing workflows must occur at the point of work, not in a separate app. A workflow system that gets used must be unobtrusive and linked to the content work at hand. This point-of-action imperative means we should split the BPM functions into atomic operations distributed across current applications, all governed by consensual workflows and terminology.

We need to embed state transitions and state designation changes into existing workflow screens. We need to look to our major content platforms (word processors, spreadsheets, content management systems, and the like) and find where we can embed simple workflow-related functionality. Some of this may be as simple as recording state transitions; others might be specific tabs or operations. Plug-ins are a proven model and can emit simple data structs recording their actions, invoked manually or automatically depending, to a REST-ful Web service linked with the content ontology. From a UI perspective, this should be done consistently in context with the host tool for all workflows. Some of these activities, such as editing or managing ontologies (knowledge graphs), tagging content, mapping content, or further refining terminology and semantics, are new tasks and not merely state changes of current tasks. These functions become a bit more complicated Web services, as we discussed in *Chapter 12*.

Based on today's standards, it would be wise to link our ontology design to some form of meta-model that would enable us to talk directly with BPMN. This notation covers the range of known and anticipated BPM and workflow activities and states.

Third-party tools allow users and analysts to inspect and modify workflows graphically and to emit their specifications in canonical forms.[3] A good design would let users and analysts examine and refine workflows directly.

What we are seeking is a framework and workflow that naturally allows us to present all existing and new content through a pipeline that extends from authoring and review to metadata assignments. Making final assignments for subject tags from the candidates and then ensuring we correctly assign all other metadata may be either eased or impeded by the actual workflows and interfaces. The trick to such semi-automatic processes is to get these steps right. Analysts need manual overrides when suggested candidate tags are not right. Sometimes new terms and entries are found when reviewing the processed content; these need to be entered and then placed into the overall knowledge graph as discovered. The process of working through steps on the tag processing screens should be natural and logical. Some activities benefit from very focused, bespoke functionality, rather than calling up a complicated or comprehensive app.

In business settings these steps need to be recorded, subject to reviews and approvals, and with auditing capabilities should anything go awry. Potential revision means there needs to be a workflow engine underneath the entire system, recording steps and approvals and enabling things to be picked up at any intermediate, suspended point. These support requirements tend to be unique to each enterprise. Thus, we favor an underlying workflow system that can be readily modified and tailored — perhaps through scripting or configuration interfaces. We also want version control systems for our knowledge graphs so that we may record, compare, and rollback changes as required.

Respect for workflows is also a first principle, expressed in two different ways. The first way is that we should not unduly disrupt existing workflows when introducing interoperability improvements. While workflows can — and should — be improved or streamlined over time, initial introduction and acceptance of new tools and practices must fit with existing ways of doing tasks to see adoption. Workers resist jarring changes to their existing work practices. The second way that workflows should be respected is the importance of being aware of, explicitly modeling, and then codifying how we do tasks. This focus becomes the 'language' of our work and helps define the tooling points or points of interaction as we merge activities from multiple disciplines in our domain. These workflow understandings also help us identify useful points for APIs in our overall interoperability architecture. These considerations provide the rationale for assigning metadata to characterize our information objects and structure, based on controlled vocabularies and relationships as established by domain and administrative ontologies.

Peirce's guidelines and KBpedia provide some unique strengths to a BPM initiative. Events, states, roles, and actions are well-characterized and structured. We have repeatedly seen the semantic technology influence in KBpedia, an essential perspective for capturing consensus and terminology related to business processes and workflows. We have put forward a 'pay-as-you-benefit' strategy for incremental testing and adoption of new scope and functionality, an approach that also fits well with implementing what may prove to be a broad, or business-wide, BPM implementation plan. Moreover, from an architectural standpoint, we have put forward a WOA design that supports atomic and distributed functionality interacting via Web services, the only approach that makes logical sense for a workflow management system.

An effective BPM system would bring tangible benefits in three ways. The first is that for us to gain efficiencies by climbing the learning curve, we must have a documented business process that we can repeat and refine. The second benefit is that we gain a basis for learning about learning. Today, knowledge gets produced. Still, we have little insight into how to do it nor how we can do it better. The third and longer-term benefit is that with a better understanding of states, actions, and events, we provide a possible entry point into real-time knowledge supervision. Processes are a Thirdness, and mediation is a dynamic process of what exists and how chance and change may affect it.

SEMANTIC PARSING

Parsing is the identification and segregation of string symbols into the constructs of a formal grammar.* A formal grammar is a set of rules for how to process these symbols, often including defined classes (lexemes) to which the processed symbols may be assigned, the aggregate of which is called the lexicon. The processing of text is like Pac-Man chewing through tokens in either left-right or right-left directions, top-down or bottom-up, by character, word or phrase, deciding at each token how to transform it or terminate. The parser might be simple, perhaps relying only on heuristics or regular expressions and seeking only to define token boundaries. The parser might be quite sophisticated and based on machine learning of the optimal methods and parameters to parse domain content for specific domain purposes.

Different NLP methods may benefit from different parsers or grammars. Some output from the parse such as tables, trees, or vectors may be suitable for different purposes or content. The tree structure, for example, is a proven storage structure for parsed documents and Web pages. Some parser generators can also effectively operate in 'reverse,' in which case they may perform as compilers (for computer languages) or syntax or grammar checkers (for languages) or theorem provers (if logic based). Thus, much research is potentially transferrable among disciplines.†

Lexical analysis is often the first stage of parsing, wherein the system 'chunks' or tokenizes the string into lexical units. For natural language understanding, the lexical constructs are parts-of-speech, word senses, sentence structure (syntax), and the like. These constructs intimately link to the formal grammar. Parsers need to sequence the string in specific ways and often rely on recursion to keep the algorithm simpler and better performing. The recursion method may thus impose other requirements on chunking order or storage or perhaps add pre- or post-processing

* The word *grammar* is derived from a Greek word meaning 'writing,' though at one time the knowledge of Latin grammar was viewed as endowing one with magical power, from which arose our word *glamour*.[4]

† In many areas of computational linguistics, care should be taken when comparing findings from the contributing disciplines.

steps. We may impose simplifications or feature reductions to keep the language analysis decidable or have it complete in acceptable time. Circumstances of use may also require us to attend to other tasks during the parse steps, including normalization, word forms, word segmentation, stemming, case adjustments, lemmatization, or sentence detection (end, beginning).

Formal grammars have different degrees of expressiveness. Once we process a natural language into a formal grammar, its reverse translation may not retain the same expressiveness, making the grammar 'lossy.' We may choose to accept some loss, since capturing the full expressiveness of natural language may require larger and more complicated formal grammars with weaker performance and greater storage.[5] Parsing and their accompanying grammars are the keys to natural language understanding. Peirce has much to offer in these areas, though toolmakers have yet to exploit parsers and grammars based on Peircean principles to any real degree.

A Taxonomy of Grammars

At the syntax level, we can classify grammars into phrase structure (or constituency) ones and dependency ones.[6] The guiding idea behind constituency grammars is that groups of words may act as a single unit, such as a noun-phrase (NP) or verb phrase (VP). Dependency parsing can express word dependencies (such as some semantic relationships) and is getting more attention because of its suitability to some forms of machine learning. Dependency parsing works well for natural languages that have free word orders (*e.g.*, Turkish, Czech). Dependency parsing can also be used to generate treebanks, which have become popular reference structures for use by tokenizers or text annotators. Example dependency grammars include word grammar, functional generative description, and link grammar.* However, the more common parsers use constituency grammars.

A formal grammar provides a set of transition rules for evaluating tokens and a lexicon of types that can build up, or generate, representative language structures. The tokens are either terminal or symbolic, with the terminal ones causing the process to continue to the next token or to stop entirely. Formal grammars act like abstract machines (or automata). Automata theory, the basis of finite-state machines, is also closely related in that an automaton is a finite representation of a formal language that may be an infinite set. Grammar with a larger lexicon of types or more sophisticated steps encourages more straightforward representations and better generalizations, including recursion, to reduce evaluation times.

Categorial grammar, a constituency grammar derived from the simply typed lambda calculus, is based on types and is built according to the principle of compositionality, wherein we understand complex expressions from the meaning of their components and the rules (grammar) of their construction. This grammar is a phrase-structure grammar, better known as a context-free grammar, in which a terminating symbol never appears on the left-hand side of a transformation step. Con-

* Nivre argues that a dependency grammar is not a grammar formalism, rather a specific way to describe the syntactic structure of a sentence.[7]

text-free languages are the theoretical basis for the syntax of most programming languages.

Since formal grammars are a branch of formal language, we can draw upon a rich mathematics literature of theory, constructs, and algorithms. In the 1960s, theoretical research in computer science on regular expressions and finite automata led to the discovery that context-free grammars are equivalent to nondeterministic pushdown automata.[8] This discovery led to the interaction of formal grammars with compiler construction. It also led to the design of deterministic context-free grammars that could be parsed sequentially by a deterministic pushdown automaton, a requirement in early programming language designs due to computer memory constraints.

Noam Chomsky was the first to formalize the idea of the hierarchical constituency with a phrase-structure grammar in 1956, which he proceeded to expand upon and develop over the ensuing decades, called the Chomsky hierarchy,[9] which splits into four types. Deterministic context-free grammars (DCFGs), an intermediate grammar in the Chomsky hierarchy, are a proper subset of the context-free grammars, which can derive from deterministic pushdown automata. As benefits, we can parse DCFGs in linear time, and a parser generator can automatically generate them.

For natural languages, practitioners favor context-free grammars, another intermediate type in the Chomsky hierarchy. Here is a sampling of the methods or grammars that have emerged from context-free grammars (CFGs):

- Affix grammar,
- Attribute grammar,
- Categorial grammar,
- CYK algorithm,
- Earley algorithm,
- Generalized context-free grammar,
- Generalized phrase structure grammar,
- Head-driven phrase structure grammar,
- ID/PL grammar,
- GLR parser,
- Lambek calculus,[4]
- Lexical functional grammar,
- LL parser,
- Minimum recursion semantics,
- Parsing expression grammar,
- Pregroup grammar,
- Phrase-structure grammar, and
- Stochastic context-free grammar.

Categorial grammars create fixed lexicons that assign a category (type) to each symbol and inference rules for what type of symbol follows, sufficient to specify a particular language grammar. The CYK algorithm is widely taught and implemented and is a good basis for understanding context-free grammar aspects.[10] Head-driven phrase structure grammar (HPSG) marks entries with a hierarchy of types. As more rules get added to HPSG, the approach takes on the form of what researchers call a construction grammar. Minimal recursion semantics is a meta-level language for describing semantic structures in a typed formalism that the authors claim is an easy way to decompose, relate and compare semantic structures.[11] In phrase-structure grammars feature sets are attribute-value pairs, where the value may be single, multiple, or complex, including lists, sets, or functions.* Another nice aspect of a feature structure is that we can represent them as a directed acyclic graph (DAG), with the nodes corresponding to the variable values and the paths to the variable names. Further, we can effectively transform every context-free grammar (CFG) into a weakly equivalent one without unreachable symbols (unprocessed tokens in the string).

Researchers strive to find a sufficiently expressive grammar, perhaps with some heuristics for rare edge cases, to capture and re-write back natural language sufficient for effective communication. It is clear that some features of languages are not context-free. It turns out, as Joshi showed for some leading-edge grammars, that we need only capture partial aspects of context-sensitivity to obtain sufficient expressivity, what he classed as mildly context-sensitive grammars. Here are some prominent options:

- Combinatory categorial grammar,
- Embedded pushdown automaton,
- Head grammar,
- Linear-indexed grammar, and
- Tree-adjoining grammar.

We may associate the elements of combinatory categorial grammar (CCG, which is grounded in combinatory logic), such as verbs or common nouns, with a syntactic 'category' that has a function with specified arguments and a type of result.[12] CCGs combine descriptive adequacy — that is, applicability to the constructions and interpretations of a wide range of diverse languages — with explanatory adequacy, in the sense of having the fewest expressions to obtain an adequate level of theoretical linguistic competence. This level of 'mildly context-sensitive grammars' is the current 'sweet spot' within the Chomsky hierarchy for trading off performance with expressiveness. The generalized linear context-free rewriting system has proven an enabler for formulating and testing new grammars at this leading edge of performance.

Not all formal grammars are generative, either. Constraint grammars are entirely rule-based, often embracing hundreds of rules. Constraint-based grammars state the rules that are disallowed, with many acting as constraint analogs to standard generative models. Functional theories of grammar try to model the way language is used in communications under the assumption that formal relations between linguistic elements are functionally motivated.

* We talked of this simple data struct in *Chapter 9*.

Computational Semantics

So, we now have a broad view of the mechanics of formal grammars and parsing, but what about the meaning of language, its semantics? We can see the processing rules and approach; we still need to understand the semantics of the chunks involved and their contribution to a representation of meaning. Computational semantics is the study of how to automate the process of constructing and reasoning with meaning representations of natural language expressions.[13] Semantic parsing breaks natural language into logical forms[14] — that is, an unambiguous artificial language — with the logic intended to express the meaning of the language components.[*] Shallow semantic parsing uses discriminative models, like recurrent neural networks, to label the roles in a sentence.

Joachim Lambek was one of the pioneers of the mathematics of sentence structure and syntax and formulated many algebraic approaches of early computational linguistics using his Lambek calculus. He acknowledged that the idea behind this approach could be traced back to Charles Sanders Peirce's ideas about valency in chemistry.[4] Lambek grammars, built using the Lambek calculus, extend basic categorial grammars. The Lambek calculus helped stabilize approaches and notations and was a forerunner to Montague grammar.

The central idea of Richard Montague's first paper in 1970 was to frame linguistic semantics as a homomorphic mapping between two algebras, one syntactic and the other semantic. In a series of three papers in the early 1970s Montague[†] fleshed out a formal theory that represents the standard theory for computational semantics for most of the last of the 20th century. We call this basis the Montague grammar (MG).[5] Montague expressed the semantics of the source into a logical form based on a theory of the semantics. He provided a functional mapping between the syntax and the logical form that preserves the structure and equivalences. While the statement of this approach seems straightforward, maintaining the homomorphism (same shape) between the forms is the tricky part.[2] The intensional logic of Montague grammars is a typed lambda calculus.[‡] Before Montague, linguists had no methods for assigning a compositional semantics to natural language syntax due to the mismatch with first-order logic. Montague's type theory represents a solution to Gottlob Frege's desire to use function-arguments as the basic 'glue' to combine meanings, a view unknown in linguistics at the beginning of the 1970s, yet now viewed as standard.[17]

Montague grammars have been a stepping stone in many different directions. One direction is that MGs presume a tree structure, which favors FSTs, HMMs, and other finite state methods of syntax analysis.[5] Another direction is to generalize into algebraic terms, making the system more functional with better information theoretics. One direction has been to combine different ideas of semantic primes as the starting lexicon.

Much of the work in semantic linguistics has focused on the commonalities be-

* For a sample detailed description see SLING, a frame-based semantic parser using a dependency grammar.[15]

† Montague's contributions came to an untimely end when he was violently murdered at age 40.

‡ This makes these grammars well suited to functional languages like Lisp.

tween human languages. One way is to use semantic primes, a basic list of primitives under which to categorize terms. Anna Wierzbicka first posed these ideas in a seminal work in 1972.[18] The universal dependencies provide shared starting points between scores of human languages. The 'universal semantic tagset' (a different concept) provides a (growing) set of cross-language primitives.[19] These initiatives show how data gathering and comparisons between human languages, made available through the Web, are remaking how computational semantics may move forward.

Three Possible Contributions Based on Peirce

Let us now weave Peirce and his potential contributions into the narrative. Consistent with the Peircean guidelines through this book, we should look to be:

- *Real* — Peirce advocated empirical truth for describing and organizing the things in the world. Definitions or arrangements based solely in the mind are psychological and not phenomenological. Hewing to a test of reality means what we retain should be true in relation to what we have already modeled, helping to ensure our methods remain consistent and coherent;
- Organized according to the *universal categories* — continuing to maintain reasoned splits into Firstness, Secondness, and Thirdness may offer some surprising keys and insights for our knowledge representations going forward;
- *Logical* — since logic is at the heart of the Peircean view. Logic fits well with the ideal of formal grammars;
- Consistent with the *logic of relations* — Peirce has already provided us with significant guidance in his identification of relations and his logical treatments of them, including algebraic notions to inform modeling;
- A good *entity-attribute distinction* — we have already pointed to the importance of separating out attributes (a Firstness) from entities (a Secondness);
- Capable of distinguishing *generals* from *particulars* — we want discrete class-level types (generals, a Thirdness) and item-level (particulars, a Secondness) ones;
- Attentive to the *sign representativeness* in Peircean semiosis — Peirce's ten sign classes (see *Table 2-2*), or even analysis of his later 28- and 66-sign classifications, are a rich target for applying mathematical or logical analysis for teasing out rules for analyzing problems;
- Reflective of the *probabilistic nature* of truth — we should favor learning models that support inductive reasoning and allow the use of probability distributions to characterize some nodes; and
- *Contextual* — in that we capture both the intensionality and extensionality of our lexemes and chose word senses based on the overlap with accompanying text. The inclusion of inference and background world knowledge support this aim.[20]

I discuss below three different approaches by which we may embrace these Peircean guidelines in whole or part. I present the approaches in relative order of complexity of implementation, starting with the simplest. We begin with Peircean part-of-speech tagging, move to machine learning, and then conclude with a dedicated Peircean grammar. We can also combine these three approaches in various ways.

#1 - Peircean POS Tagging

The first and most direct incorporation of Peircean themes likely resides in relating these constructs to off-the-shelf part-of-speech taggers. One quick approach might be to map an existing schema to the KBpedia components, which we have already organized in a Peircean manner. Chen, the originator of the E-R modeling ideas, understood the entity-attribute split well. I have taken Chen's mapping of word senses [21] and related it to existing KBpedia components:

Word Sense	*KBpedia Component*
common noun	concept / type / entity /event
proper noun	entity/event
transitive verb	relation
intransitive verb	attribute
adjective	attribute
adverb	attribute (property)

Table 16-1: Simple POS Mapping

These senses map pretty well but lack consideration of sub-types within entities, external relations or types. They also neglect many of the 'gluing' parts-of-speech such as determiners, conjunctions or prepositions.[22] Some modifications to an E-R model approach might be undertaken to embrace the full structure of languages better as found in some reference tagsets, but that is a demanding, manual task. Ninio, in a recent review informed by Peirce, also put forward an approach to syntactically label parts of speech.[23]

We need to go deeper into Peirce's ideas about signs, language, and grammar to understand how a Peircean approach to POS tagging might better proceed. Peirce had strong interests in word categories, more from a semantic than syntactical perspective, with original ideas about common nouns, proper nouns, pronouns, verbs, and prepositions.[24] Peirce understood a sentence as a formal proposition split into two fundamental parts, the subject and the predicate (1902, CP 2.318).* In a formal proposition, the subject is definite. Subjects often begin as indefinite individuals (such as 'selectives',[25] *e.g.*, *some person*), proceed as better understood and characterized into a definite individual (a 'proper noun,' *e.g.*, *Jimmy Johnson*), and then may be related to a type, a definite general (a 'common noun,' *e.g.*, *football coach*). (1905, MS

* By formal proposition I mean a sentence in the indicative mood; "for a proposition is equivalent to a sentence in the indicative mood" (1903, CP 2.315), for which Peirce was mostly concerned. Contrast this to the other moods (1893, CP 2.291) or 'quasi-propositions', see below.

280:41) However, common nouns are not universal,[26] with proper nouns providing the ultimate subjects.[27] Some predicates bring with them the need to also specify a direct object as a complement to the intended subject, as in 'Cain killed Abel.' (1899, CP 2.230) Other sentence constructions may also require multiple subjects through the use of conjunctions ('Bob and Mary went shopping') or triadic relations ('Bob donated a scholarship to Mary').

It is these kinds of constructions that help instruct what Peirce meant by a predicate, or what he termed a *rheme*. A rheme is an 'unsaturated' term (1902, CP 2.317), meaning it stands as the function within a propositional phrase that lacks (is 'blank') a subject. Here is Peirce's definition for verb:

> "A *verb*, being understood in a generalized sense, may be defined as something logically equivalent to a word or combination of words, either making a complete proposition, or having certain *blanks*, or quasiomissions, which being filled each with a proper name, will make the verb a complete proposition." (1896, NEM 4:278)

Peirce goes on to say that "The places at which lines of identity can be attached to the verb I call its *blank subjects*." (1898, NEM 4:338)

This idea of *blank subjects*, and the role of the index in relation to them, is one of Peirce's pivotal perspectives. As Nöth notes, "indexical signs had traditionally not been associated with the concept of representation, and indeed, the terminological tradition had been to subsume only iconic and symbolic signs under this term."[28] Peirce helps show and generalize the range of relations between things, between subjects and predicates, which indicate ranges of determinacy or selectiveness. We see, for example, that we may characterize clauses, verb and noun phrases, prepositions, indicatives,* adverbs, and adjectives according to their indexicality and whether the subject is determinant.

Of course, the initial split of sentences into subject and predicate masks the fact their syntax may be somewhat complex. Peirce applies the same logic, though, to noun phrases and verb phrases as well as to other constructs he calls 'quasi-propositions.' This construct, which Peirce named a dicisign or dicent, is information-bearing and adds further characterization to its subject. When decomposed, a dicisign acts like a value pair with its two signs, like a full proposition, providing a function sign (predicate) and a denotation sign (subject). The subject may itself be an index or indeterminate, one of the reasons why Peirce called them 'quasi-propositions.' Like Peirce's viewpoint of the *breadth* of information, the *dicent sinsign* points to more characteristics, or attributes, of the intended subject. Like the *depth* of information, the *dicent indexical legisign* points to subsumption (' ___ *is a man*' implied by the *man* common noun) or external relations. Hilpinen provided some of the first detailed analysis of how Peirce viewed the proposition.[29]

Table 16-2 combines these insights to characterize Peirce's ten signs linguistically. Note that the order in this table changed from *Table 2-2* where the dominant ordering was qualisign-sinsign-legisign (consistent with Peirce's 1903 ordering in his *Syllabus*

* Such as *this, that, something, anything.*

[EP 2:294-295) to rheme-dicisign-argument (consistent with Peirce's 1904 ordering in a letter to Lady Welby [CP 8.341]):

Sign Name (redundancies)	*Comments*	*KBpedia*	*POS**
(Rhematic Iconic) *Qualisign*	onomatopoeic,[30] ideophones, uncountable (mass) nouns (?)	abstractives	
(Rhematic) *Iconic Sinsign*	indefinites, conjunctives, disjunctives	attributes	N, VI, ADJ, ADV
Rhematic Indexical Sinsign	direct experience, relations	external relations	DT, N,VT
(Rhematic) *Iconic Legisign*	metaphors, puns, analogies, diagram-like, genres (?)	associations	N, VI, ADJ, ADV
Rhematic Indexical Legisign	demonstratives, pronouns, proper names	particulars	N, VI
Rhematic Symbol (Legisign)	common nouns	types	N, AUX
Dicent (Indexical) *Sinsign*	'quasi-propositions' for adjectives, adverbs, modifiers (in depth)	attributes	NP, VP, ADJP, ADVP
Dicent Indexical Legisign	'quasi-propositions' for information in breadth	external relations, subsumption, is-a	NP
Dicent Symbol (Legisign)	proposition (full), sentence	OPEN	NP + VP
Argument (Symbolic Legisign)	multiple sentences	graph measures	---

Table 16-2: Peirce's Ten Signs for KR Relation to Linguistics

The sense that emerges is that Peirce's strong links to information and representation mean there is much of value in Peirce's linguistic views to KR and knowledge management. While the information in this section could be used to set different bases for labeling syntactic categories, it may be better to logically continue the effort to develop a tokenizer more attuned to Peirce's unique views, as I discuss next.

#2 - Machine Learning Understanding Based on Peirce

The index acts as a reference to the object, ultimately representing an individual thing (including individual collections). (EP 2:407) The interpretant must have some previous (or 'collateral') acquaintance with the object to identify that individual thing, what Peirce termed '*collateral observation*.'† By this term, Peirce meant "previous acquaintance with what the Sign denotes." (EP 2:494) Nothing of this observation is psychological since the interpretant contributes no part to the observation. The collateral observation plays a parallel role to context but is not the same. Collateral observation is central to disambiguation since presently observed characters may be compared with previous observations to separate out identities. Collateral observa-

* I view these assignments as provisional. There is not much in the literature (or Peirce directly) on these assignments. I anticipate further research to refine these assignments somewhat.

† Peirce also termed this *collateral experience*, *collateral information*, and *collateral acquaintance*.

tions extend beyond the boundaries of the sentence.

Context and most arguments also extend beyond the boundaries of the sentence.* If we are to hope for acceptable levels of *natural language understanding* through KR formalisms, we must also tackle these issues of collateral observation, context, and the reasoned argument. To get at these aspects we perhaps want to combine some form of semantic parser, better attuned to our organization of KBpedia by the universal categories and types, and machine learning, possibly leveraging *knowledge supervision*, as we discussed in *Chapter 4*.

A semantic parser requires us to formulate a semantics, including language construction, and then to learn how to apply it to new content with acceptable computational times.[14] Sarbo and Farkas suggest a rather simple parsing method grounded in their interpretation of Peirce.[30] I agree with the authors that we want simple models because most formal models of natural language are too complex. I also like their approach using a pushdown automaton, which is more capable than a finite-state machine. However, I do not agree with their grammar basis or some of their construction rules. We have a different mindset and structure in KBpedia in the universal categories and typologies.

A couple of approaches look promising for next steps. One of the approaches, combinatory categorial grammar (CCG), we introduced above. The other approach, Lambek categorial grammar (LCG) is closely related. CCG is an efficiently parseable, yet linguistically expressive grammar formalism. Because of its strong lexicon approach and suitability to types, CCG should be a good match with the typology design of KBpedia. Researchers have developed a probabilistic CCG from question-answer pairs using supervised learning from ontologies and knowledge bases.[31] As shown for transitive verbs, we can substitute meaning vectors as the learning basis.[32] Perhaps more promising is a tensor-based semantic framework that can be "seamlessly integrated" with CCG for a "practical, type-driven compositional semantics based on distributional representations."[33] Edward Grefenstette's thesis provides excellent guidance on how to relate distributional representations of meaning to CCG.[34]

Another useful aspect of KBpedia is the availability of a text corpus (largely from Wikipedia) for all of the reference concepts in the system. By leveraging this content, we can create distributional representations that enable us to overcome fixed-pair inputs (such as question-answer) used to train many of these CCGs. This additional distributional component improves generality beyond the fixed input vocabulary used in training, making it more suitable for open-vocabulary applications.[35] Lastly, CCGs warrant testing against KBpedia due to the availability of open source implementation kits. OpenCCG is perhaps the best known, with helpful online tutorials.

Though invented before CCGs and overlooked for a period, LCGs provide a simpler and more transparent mapping between phrase-structure trees, dependency structures, and semantic terms. [36] CCGs, in practice, have tended to need a large number of non-categorial rules, making them harder to understand and less generalizable.[36]

One trend we see in computational linguistics is to combine logical and statistical

* Of course, it is possible to write out full syllogisms in the confines of a single sentence, but most often arguments are made over multiple sentences.

approaches to natural language.* Logical, or compositional, approaches relate syntactical phrases to the meanings of their parts and how they are combined. These are the traditional approaches of semantic parsers to map messages to logical forms, which lend themselves to dealing with inference, ambiguity, and vagueness. On the other hand, statistical approaches, including machine learning, focus more on the individual word and phrase meanings or broader notions of content (and context) beyond the sentence.[39] What is exciting about a combined approach is that we can look at the compositional and semantic aspects of our language, mapped into the categorial perspectives of Peirce's logic and semiosis, and then convert those formalisms to distributions over broad examples provided by KBpedia's knowledge.

#3 - Peirce Grammar

At a more speculative level, we can look to going to the heart of the matter: fully adopting Peirce's views on logic and relations regarding how we specify our grammars. Full adoption is not such a wild idea since there have been attempts and probes around a 'Peircean grammar' for at least a couple of decades. The basic advantage of this approach is that we do not need to shoehorn Peircean ideas into existing approaches, but are free to set up a clean infrastructure from scratch.

Patrick Suppes was one of the first to question the traditional approach of translating sentences into the formal notation of predicate logic.[40] He observed that inference in predicate logic bore little resemblance to the informal reasoning in English. He began to explore what he called *extended relation algebras*, which were a model-theoretic semantics for English that used neither quantifiers or variables, but only constants on operations on sets and relations.

Chris Brink, in his 1978 thesis, was explicit about the influence of Peirce.[41] He noted that Peirce's first 1870 paper on the "Logic of Relatives"[42] was instrumental in guiding his thinking. Peirce classified logical terms into three classes — absolute terms, (simple) relative terms, and conjugatives — which correspond roughly to monadic, dyadic, and triadic predicates.[43] Within a decade Brink and his students were referring to this approach as the 'Peirce algebras,' a term which has stuck. One of the basic operations was the 'Peirce product,' R:A, which is the set of all elements related by R to some element in A (a basic matrix algebra). Relatives, as dyadic relations, can be represented algebraically rather than by conventional model theoretics. We can perform arithmetic over the individual identities. Boolean modules formalize the calculation of sets interacting with relations via the Peirce product. This approach enables us to treat the system as a two-sorted algebra (Boolean algebra with multiplication via the Peirce product). A two-sorted algebra makes explicit the implicit relation of concepts and roles, a concern for early KR languages for AI.

Eventually, this algebra was shown able to express the semantics of reasoning over sets, including for subsumption relations.[44] Using an algebraic approach to reason over sets becomes simpler since equations are sufficient to capture first-order

* One genesis of this grand synthesis is a 2010 paper by Coecke *et al.*, "Mathematical Foundations for a Compositional Distributional Model of Meaning,"[37] first unveiled in 2008.[38]

reasoning using the calculus of relations for unions, intersections, and complements. De Rijke provided proof for full Peirce algebras in his 1993 thesis and showed the link with dynamic modal logics.[45] This confluence of work naturally led to efforts to try to find a grammar to match this algebra.

Michael Böttner defined a 'Peirce grammar' in 2001 and applied it to natural language.[46] It is a context-free grammar. The basic idea of the grammar is a direct one using Peirce algebra. Rather than translating the semantics first to a set-theoretic metalanguage, the Peirce grammar uses only algebra with the equation sign '=' as the single predicate. A Peirce grammar allows computations on strings directly rather than to variables or a pre-computation of an intermediate representation. Böttner introduced tree structures to handle more efficiently the inherently 'flat' nature of a Peirce algebra representation (useful for programming languages; it may be less so for natural language sentences, which are shorter). A Peirce grammar can support references outside of the sentence, again as a function of storage design, attractive for context. Böttner presents a strong definition of a Peirce grammar for English (see his Table II). While the approach neglects some nuances, the approach does appear to handle the ideas of context and language fragments (such as clauses) in a computationally efficient manner.

Hans Leiß later took up some of the weaknesses and provided some extensions to overcome them.[47] Leiß noted that prior Peirce grammars had only modeled extensional aspects of natural language. It was unclear how to handle intensional aspects (attributes) or verbs with propositional arguments. The 'trick' of coding linguistic strings directly means the unit boundaries (sentence, paragraph, arbitrary window) should be finite; the limits have not been tested. Leiß raises questions about whether and how we should handle noun phrases.* On the other hand, the approach does not use variables. We can construct meanings from a few basic ones with equality and subsumption capturing the relations between sets and their relations. Leiß, as well, looks to the tree structure to provide a more tractable approach to the inherent 'flat' structure of a pure Peirce grammar. As Leiß concludes:[47]

> "Peirce grammar differs from other grammatical theories in that meanings are first-order objects, abstract sets and relations, which can only be composed by algebraic operations. Extended Peirce grammar adds a further sort of meanings, finite trees of sets and relations, from which constituent meanings can be extracted. These 'second-class' values have no ontological motivation—they only serve as intermediate stages in the evaluation of sentences, allowing us to give the context of an expression an access to the meaning *constituents* of the expression." (p. 162)

From there, for more than ten years,† the trail goes cold. Relational grammars, in general, have gone out of favor. It very well may be fundamental limits exist with relational grammars, or particularly Peirce grammars, that relegate them to a minor footnote in the history of computational linguistics. However, I suspect that Peirce grammars, as they may evolve, may yet prove a seminal player in that history.

* We are also missing a design or approach to compositionality.

† Leiß's publication is dated in 2009, but based on a conference paper presented in 2005.

COGNITIVE ROBOTICS AND AGENTS

Robotics is a potential testbed for Peircean ideas about representation and KBpedia. One reason, of course, is an economic demand for greater autonomy combined with cognition. Advances tend to appear where the most imperative resides. A Peircean approach will also aid robotics designers, and much in the ideas about representation will benefit robotics, particularly in the interface with cognitive systems. Further, while language is symbolic, cognition and understanding are more. For the idea of Thirdness to become a general, it must pass the threshold of the habitual as Peirce explains the matter. An emphatic 'Fire!' is five symbols combined into a string symbol on a page, but shouted in a dark theatre with an intonation that signals real fear, invokes immediate action. There is nothing to 'think through' before acting, though that starts immediately as well.

Cognitive robots embrace the ideas of learning and planning and interacting with a dynamic world. When combined with mobility and perception sensors, this leads to greater autonomy. When combined with speech recognition and natural language understanding (NLU), we can instruct the robot by voice commands or interact with it as a virtual agent for Q & A or knowledge assistance. Over time, researchers have tested and designed various cognitive architectures for integrating cognitive and robotic functions, with a preference for a modular design with generic interfaces.[48]

Time coordination for how long it takes modules to process their tasks require trade-offs in expressiveness; simpler and more abstract representations appear best. We also see open-source, modular robot languages and operating systems emerge (such as the Robot Operating System, ROS), the Robobrain knowledge engine initiative and even relatively affordable autonomous robot platforms (such as iCub or ROBOTIS OP2) emerge. Cognitive robotics promises to improve our baseline understanding of knowledge representation. Perception of and interaction with the external world are integral to sign communications. If we are ever to approach anything like true natural language understanding, then we need to incorporate all of the universal categories in our reality. Autonomous, cognitive robots are the anvil upon which these understandings may get hammered out.

Lights, Camera, Action!

Peirce's semiosis is not consistent with traditional computational views of cognition, which are a variation of input-output models. How the symbol gets interpreted is neglected by the traditional view.[49] Peirce's semiosis more closely represents the theory of embodied cognitive science, which differs from the tradition in pursuing three goals. These goals are to elevate the importance of the body as an explanation for various cognitions, to understand the body as a contributor to cognition, and to broaden our view of how agents use the environment to affect cognition (mood lighting, staging, arranging and positioning).[50]

Similar to our example of shouting 'Fire!,' awareness of the environment is an essential factor in cognition. The homunculus argument of the little man interpreting things in our heads must be grounded in some external reality to keep that interpretation from cascading into an infinite regress. This 'interpretation solely in the mind' is the fallacy in Descartes' worldview, as well as in the traditional view of cognition. Successful cognition also requires learning, and epigenetic robots depend on perceptions or kinesthetics (learning from physical actions) for the new information from which to learn. The forming of symbols from that learning is one way to achieve a certain Thirdness. Further, as long as we look at knowledge as only symbolic, we will miss the importance of Firstness and Secondness. Everything has its place.

Peirce indeed acknowledged the unconscious reaction and the role of instinct in signs and how we interpret them. Peirce believed in an unconscious aspect of the mind (1903, EP 2.188; 1903, CP 5.108; 1882, CP 7.64; *c.f.*, 1902, CP 7.363-367). Formal reasoning, the sphere of theory and analytics and logic, is part of the conscious mind, but the realm of action and pragmatic knowledge is a function of the unconscious.

> "Reasoning, properly speaking, cannot be unconsciously performed. A mental operation may be precisely like reasoning in every other respect except that it is performed unconsciously. But that one circumstance will deprive it of the title of reasoning. For reasoning is deliberate, voluntary, critical, controlled, all of which it can only be if it is done consciously. An unconscious act is involuntary: an involuntary act is not subject to control; an uncontrollable act is not deliberate nor subject to criticism in the sense of approval or blame. A performance which cannot be called good or bad differs most essentially from reasoning." (1903, CP 2.282)

Peirce held belief, which we saw in *Chapter 2*, as an important aspect of knowledge that occurs mostly in the unconscious. (1905, EP 2:336; 1905, CP 5.417) Habitual stuff and reactive actions are part of common sense and not (generally) part of consciousness. However, the informal 'reasonings' in the unconscious are often more reliable than conscious reasoning and logical inference:

> "Association is the only force which exists within the intellect, and whatever power of controlling the thoughts there may be can be exercised only by utilizing these forces; indeed, the power, and even the wish, to control ourselves can come about only by the action of the same principles. Still, the force of association in its native strength and wildness is seen best in persons whose understandings are so little developed that they can hardly be said to reason at all. Believing one thing puts it into their heads to believe in another thing; but they know not how they come by their beliefs, and can exercise no control over the inferential process. These unconscious and uncontrolled reasonings hardly merit that name; although they are very often truer than if they were regulated by an imperfect logic, showing in this the usual superiority of instinct over reason, and of practice over theory. They take place like other mental suggestions according to the two principles of similarity and connection in experience." (1886, CP 7.453)

I think two implications arise from Peirce's observations. First, we begin to unveil a bit of the role of knowledge bases as 'belief' bases insofar as they make direct assertions. We can know that balls are round and squares have four equilateral sides and can act on these assertions without further analysis. Second, 'instinct,' as ephemeral as it is to define, plays an essential role in guiding actions. Still, what does Peirce mean by 'instinct,' one of his most common descriptors?

Peirce first sees two sources of instinct, one innate or biological, the result of evolution, what he calls *inherited*, and the other one of infant training and inculcation, what he calls *traditional*. Peirce notes that instincts may change when circumstances change, but that is rare since circumstances hone our instincts over generations of trial and error. Peirce splits his views of logic into a *logica docens*, the logic of theory and study, and a *logical utens*,[51] the internal logic of practice as influenced by instinct. The first logic is that of the scientist, the second that of the practical actor.

> "If an action, although complicated, has very often to be performed, and is almost always performed in nearly the same way, it frequently happens that we have an instinct for performing it. The action of walking is an example; the action of throwing a stone is another. Now instinct is remarkable for its great accuracy, as well as for its adaptedness to its purpose; and it would usually be unwise in the extreme to attempt to perform such an act under the guidance of theory; for theories have to be studied very long and very deeply before they can be entirely freed from error; and even then the application of them is laborious and slow." (n.d., NEM 4:187)

Peirce did not view instinct as inferior to formal reasoning while noting that "action in general is largely a matter of instinct." (1905, CP 5.499) He saw that "we all have a natural instinct for right reasoning." (1902, CP 2.3)

> "If so, in what respect do you hold reasoning to be superior to instinct? Birds and bees decide rightly hundreds of times for every time that they err. That would suffice to explain their imperfect self-consciousness; for if error be not pressed upon the attention of a being, there remains little to mark the distinction between the outer and the inner worlds." (1902, CP 2.176)

> "Of excessively simple reasonings a great deal is done which is unexceptionable. But leaving them out of account, the amount of logical reasoning that men perform is small, much smaller than is commonly supposed. It is really instinct that procures the bulk of our knowledge; and those excessively simple reasonings which conform to the requirements of logic are, as a matter of fact, mostly performed instinctively or irreflectively." (1902, CP 2.181)

Peirce saw that instinct and abduction are linked.* (1903, CP 5.171) While Peirce held that pragmatism is a conscious discipline (and thus in the realm of *logica docens*), there may be instinctual aspects of how we conduct it. We often instinctively screen through the multiplicity of abductive options to select those for more expensive inductive attention. We need to evaluate options and potential outcomes based on practical measures and instincts. Our conscious reasonings need to incorporate an inspection and role for instinct.

Practical implications for robotics arise from this discussion. We can envision

* Note that Peirce specifically excludes consideration of instinct in the scientific method and its quest for truth, since all assumptions should be open to question. Pragmatism, however, adds action and instinct to the equation.

sub-systems that deal with direct knowledge — based on the ABox and perhaps direct typing — tied more closely to the perceptual sensors and kinesiology of the agent. These subsystems may be linked more closely to instinctual actions such as resource acquisition, risk avoidance, and protection. Perhaps this perspective offers insight into how to monitor the external environment while all inputs are within expected parameters versus outliers or disruptions that may need to trigger more analytic modules. It would also seem that instinctual sub-systems may be more attractive areas for rules-based approaches or unsupervised machine learning. The underlying idea of instinct is what is the best way to act under given contexts and events.

Cognitive versus reaction sub-systems also require different time demands and impose different time delays. Robot information architectures need to define modules and optimize effectiveness-performance trade-offs. The integration of NLU requires complements to perceptions and actions, a current area of active interest.[52] The conventional perspective is to ask how we can better import existing knowledge representation and systems into cognitive robots. However, perhaps we need to pose that question the other way around.

> "Wisdom lies in nicely discriminating the occasions for reasoning and the occasions for going by instinct. Some of my most valued friends have been almost incapable of reasoning; and yet they have been men of singularly sound judgment, penetrating and sagacious." (1903, CP 7.606)

Grounding Robots in Reality

One way to avoid the homunculus' infinitely regressing explanations is to 'ground' our symbols into some base truth, called the 'symbol grounding problem.' Grounded symbols no longer have free variables and become the 'indecomposable' primitives of the representation. The implication is that higher-order concepts are derivations of lower-level concepts until the concepts can no longer be divisible. Ultimately, for cognitive robots, the processing of natural language, be it from commands or interacting with humans, must be part of this grounding. It may take the form of a semantic model underlying both language and robot commands that is also related to robotic perception and actuation; see combinatory categorial grammars[52] as mentioned in the prior use case. Cangelosi sees the symbol grounding problem in similar terms, where the questions of perception and action and how they are represented mentally is a core issue in cognitive robotics.[53] Deb Roy sees the representation more broadly, embracing the idea of symbols as included in semiosis based on Peircean concepts.[54] He also wants to specifically relate "sensing and motor action to words and speech acts."

While essential considerations, some argue that the question of grounding goes well beyond mere representation.[55] The nature of the question is evolving to one of *meaning** and how that relates to *grounding*. That is because for a cognitive robot to

* Note that *meaning* has many connotations including existential, linguistic, philosophical, psychological, semiotic, of life, and others. Our use embraces all of these senses.

formulate new knowledge and symbols, there must be some guiding principles or functions that go beyond a representative grounding. Some mechanism must exist for emergence or how incorporation of new information may lead to new actionable knowledge. Roy, again, tries to get at this question by posing a split of 'meaning' into *referential*, *functional*, and *connotative* forms. He prefers a simple base language, modeled on that used by young children.[56] Stanton believes we should try to mimic the evolution of the brain and include intrinsic 'value systems' in autonomous robots to process novel experiences.[57] Stanton nibbles around the edges for how Peircean concepts may contribute to this task. We have also tended to overlook the adaptability and emergent properties of symbol systems for robotic intelligence.[58] Ricardo Gudwin, steeped in Peircean viewpoints, takes the question to a different level when he notes that we do not have a symbol grounding problem, yet one of grounding to the icon.[59] Icons are a representation of Firstness and perhaps better intimately tied to the inputs of sensors in robots.

No matter how framed, KBpedia provides three solid contributions to the grounding problem. First, of course, it provides a complete and coherent view of representation, knowledge, actions, events, and relations. Second, KBpedia as a reference knowledge graph grounds the system ultimately in Firstness (monads), Secondness (particulars), and Thirdness (generals) no matter where we start the inspection. Third, we construct KBpedia with multiple knowledge bases that can provide the reference base for both analytic and instinctual purposes and tests. The structural recursion and richness of KBpedia structure appear an excellent fit for cognitive robotic architecture and purposes.

Robot as Pragmatist

As crucial as symbol grounding is, I think we still may be missing the pivotal importance of robotics to knowledge-based artificial intelligence. Up to this point, we have framed the challenge as one of getting AI advances — including ideas of representation and meaning — *into* robots. What of the other way around?

What this short survey has shown us is that robots may bring their own contributions to these questions. We have seen how important it is to integrate the dimensions of perception and action into a cognitive processing robot. We recognize both analytic (cognitive, thinking) tasks and activities more dominated by instinct and kinesthetic action. Cognitive robots demand that we deal with the challenges of integration, coordination, and choreography. I think it fair to observe that doing KBAI in a purely symbolic, unembodied state is likely to provide an incomplete testbed for knowledge, cognition, and learning. Human intelligence evolved in a mobile, interactive environment. We may need to embed artificial intelligence in dynamic, physical contexts to approximate similar capabilities.

Parisi *et al.* argue how multimodal representations can improve the robustness of recognizing actions, action-driven perception, sensory-driven motor behavior, and human-robot interaction.[60] Robot vision systems are providing a different perspective on how we need to represent best things like shapes, vectors, and objects. Brain studies show that 'what' and 'where' have separate recognition areas, lending credence to the need for modularity.

Cognitive robots promise to help us improve our knowledge representations and AI efforts. Cognitive robots may be the drivers for better capabilities in planning, coordinated action-cognition responses, human-robot interactions, and learning to perform an ever-growing list of functions in real-time settings. Solving the competing demands for cognitive robots can only make us more pragmatic.

Chapter Notes

1. Hepp, M., Leymann, F., Domingue, J., Wahler, A., and Fensel, D., "Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management," *e-Business Engineering, 2005*, Beijing, China: IEEE, 2005, pp. 535–540.
2. Smith, F., and Proietti, M., "Ontology-based Representation and Reasoning on Process Models: A Logic Programming Approach," *arXiv:1410.1776 [cs]*, Oct. 2014.
3. See, for example, the open-source Yaoqiang BPMN editor (http://bpmn.sourceforge.net/).
4. Lambek, J., *From Word to Sentence: A Computational Algebraic Approach to Grammar*, Polimetrica sas, 2008.
5. Kornai, A., *Mathematical Linguistics*, London: Springer, 2008.
6. This split somewhat reflects a similar one for *discriminative* versus *generative* machine learning models. Discriminative models, also called conditional models, are a class of models used in machine learning for modeling the dependence of unobserved (target) variables *y* on observed variables *x*. Example discriminative models include support vector models (SVM), conditional random fields (CRF), neural networks (xNN), linear regression, maximum entropy Markov, and random forests. Generative models use algorithms to try to reconstruct how the original data was generated, often through probabilistic means. Example models include hidden Markov models (HMM), naive Bayes, generative adversarial networks (GANs), Gaussian mixture model, and other types of the mixture model.
7. Nivre, J., *Dependency Grammar and Dependency Parsing*, Växjö University, 2005.
8. See https://en.wikipedia.org/wiki/Deterministic_context-free_grammar.
9. Also known as the Chomsky–Schützenberger hierarchy.
10. Lange, M., and Leiß, H., "To CNF or Not to CNF? An Efficient yet Presentable Version of the CYK Algorithm," *Informatica Didactica*, vol. 8, 2009, pp. 1–21.
11. Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A., "Minimal Recursion Semantics: An Introduction," *Research on Language and Computation*, vol. 3, Jul. 2005, pp. 281–332.
12. Steedman, M., "A Very Short Introduction to CCG," *Unpublished draft note*, Nov. 1996, p. 8.
13. Bos, J., "A Survey of Computational Semantics: Representation, Inference and Knowledge in Wide-Coverage Text Understanding," *Language and Linguistics Compass*, vol. 5, 2011, pp. 336–366.
14. Liang, P., "Learning Executable Semantic Parsers for Natural Language Understanding," *arXiv:1603.06677 [cs]*, Mar. 2016.
15. Ringgaard, M., Gupta, R., and Pereira, F. C. N., "Sling: A Framework for Frame Semantic Parsing," *arXiv:1710.07032 [cs]*, Oct. 2017.
16. Hobbs, J. R., and Rosenschein, S. J., "Making Computational Sense of Montague's Intensional Logic," *Artificial Intelligence*, vol. 9, 1977, pp. 287–306.
17. Partee, B. H., "Montague Grammar," *International Encyclopedia of the Social and Behavioral Sciences*, N.J. Smelser and P.B. Bates, eds., Oxford: Pergamon/Elsevier Science, 2001, p. 7 pp.
18. Wierzbicka, A., *Semantics: Primes and Universals*, Oxford University Press, UK, 1996.

19. Abzianidze, L., and Bos, J., "Towards Universal Semantic Tagging," *arXiv:1709.10381 [cs]*, Sep. 2017.

20. Hassan, S., "Measuring Semantic Relatedness Using Salient Encyclopedic Concepts," Ph.D., University of North Texas, 2011.

21. Chen, P. P.-S., "English, Chinese and ER Diagrams," *Data & Knowledge Engineering*, vol. 23, 1997, pp. 5–16.

22. Tokenizers and POS taggers, plus any reference tagsets employed, should be attentive to syntax that is declinable (noun, pronoun, verb, adverb) The indeclinable terms (proposition, conjunction, interjection, particles, modals) are less of a problem since only single terms are required. Declensions of tense, case, plurality or gender are very important topics in some languages, though I do not speak further of it here.

23. Ninio, A., "Learning a Generative Syntax from Transparent Syntactic Atoms in the Linguistic Input," *Journal of Child Language*, vol. 41, Nov. 2014, pp. 1249–1275.

24. Nöth, W., "Charles Sanders Peirce, Pathfinder in Linguistics," *Digital Encyclopedia of Charles S. Peirce*, Sep. 2000.

25. A "selective" (1903, CP 4.408) is an indeterminant individual such as indicated by selective pronouns (any, every, all, no, none, whatever, whoever, everybody, anybody, nobody) or particular selectives (some, something, somebody, a, a certain, some or other, one) (1903, CP 2.289).

26. Peirce did not hold the *common noun* to be a universal POS (part-of-speech). He states, "I do not regard the common noun as an essentially necessary part of speech. Indeed, it is only fully developed as a separate part of speech in the Aryan languages and the Basque, -- possibly in some other out of the way tongues." (1904, CP 8.337).

27. Pietarinen, Ahti-Veikko, "Peirce's Pragmatic Theory of Proper Names," *Transactions of the Charles S. Peirce Society*, vol. 46, 2010, p. 341.

28. Nöth, W., "Representation and Reference According to Peirce," *International Journal of Signs and Semiotic Systems*, vol. 1, Jul. 2011, pp. 28–39.

29. Hilpinen, R., "On CS Peirce's Theory of the Proposition: Peirce as a Precursor of Game-Theoretical Semantics," *The Monist*, vol. 65, 1982, pp. 182–188.

30. Sarbo, J., and Farkas, J., "A Peircean Ontology of Language," *International Conference on Conceptual Structures*, Springer, 2001, pp. 1–14.

31. Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L., "Scaling Semantic Parsers with On-the-Fly Ontology Matching," *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington: Association for Computational Linguistics, 2013, pp. 1545–1556.

32. Clark, S., "Type-Driven Syntax and Semantics for Composing Meaning Vectors," *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*, 2013, pp. 359–377.

33. Maillard, J., Clark, S., and Grefenstette, E., "A Type-Driven Tensor-Based Semantics for CCG," Association for Computational Linguistics, 2014, pp. 46–54.

34. Grefenstette, E., "Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics," Ph.D., Balliol College, University of Oxford, 2013.

35. Gardner, M., and Krishnamurthy, J., "Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge," *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, Association for the Advancement of Artificial Intelligence, 2017, pp. 3195–3201.

36. Fowler, T. A. D., "Lambek Categorial Grammars for Practical Parsing," Ph.D., University of Toronto, 2016.

37. Coecke, B., Sadrzadeh, M., and Clark, S., "Mathematical Foundations for a Compositional Distributional Model of Meaning," *Linguistic Analysis*, vol. 36, Mar. 2010, pp. 345–384.

38. Coecke, S. C. B., and Sadrzadeh, M., "A Compositional Distributional Model of Meaning," *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, Oxford University Press, , pp. 133–140.

39. Liang, P., and Potts, C., "Bringing Machine Learning and Compositional Semantics Together," *Annual Review of Linguistics*, vol. 1, 2015, pp. 355–376.

40. Suppes, P., "Direct Inference in English," *Teaching Philosophy*, vol. 4, 1981, pp. 405–418.

41. Brink, C. H., "The Algebra of Relations," Ph.D., University of Cambridge, 1978.

42. Peirce, C. S., “Description of a Notation for the Logic of Relatives, Resulting from an Amplification of the Conceptions of Boole’s Calculus of Logic,” *Memoirs of the American Academy of Arts and Sciences*, vol. 9, 1870, pp. 317–378.

43. Brink, C., “The Algebra of Relatives,” *Notre Dame Journal of Formal Logic*, vol. XX, Oct. 1979, pp. 900–908.

44. Brink, C., and Schmidt, R. A., “Subsumption Computed Algebraically,” *Computers & Mathematics with Applications*, vol. 23, Jan. 1992, pp. 329–342.

45. de Rijke, M., “Extending Modal Logic,” Ph.D., Universiteit van Amsterdam, Institute for Logic, Language and Computation, 1993.

46. Böttner, M., “Peirce Grammar,” *Grammars*, vol. 4, 2001, pp. 1–19.

47. Leiß, H., “The Proper Treatment of Coordination in Peirce Grammar,” *Proceedings of FG-MoL 2005*, Edinburgh, Scotland: 2009, pp. 149–166.

48. Scheutz, M., Harris, J., and Schermerhorn, P., “Systematic Integration of Cognitive and Robotic Architectures,” *Advances in Cognitive Systems*, vol. 2, Dec. 2013, pp. 277–296.

49. Steiner, P., “CS Peirce and Artificial Intelligence: Historical Heritage and (new) Theoretical Stakes,” *Philosophy and Theory of Artificial Intelligence*, Springer, 2013, pp. 265–276.

50. Shapiro, L., “The Embodied Cognition Research Programme,” *Philosophy Compass*, vol. 2, Mar. 2007, pp. 338–346.

51. Chiasson, P., “Logica Utens,” *Digital Encyclopedia of Charles S. Peirce*, Jan. 2001.

52. Matuszek, C., Herbst, E., Zettlemoyer, L., and Fox, D., “Learning to Parse Natural Language Commands to a Robot Control System,” *Experimental Robotics*, Springer, 2013, pp. 403–415.

53. Cangelosi, A., “Solutions and Open Challenges for the Symbol Grounding Problem,” *International Journal of Signs and Semiotic Systems*, vol. 1, 2011, pp. 49–54.

54. Roy, D., “Semiotic Schemas: A Framework for Grounding Language in Action and Perception,” *Artificial Intelligence*, vol. 167, Sep. 2005, pp. 170–205.

55. Williams, M.-A., “Representation = Grounded Information,” *Trends in Artificial Intelligence*, T.-B. Ho and Z.-H. Zhou, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 473–484.

56. Roy, D., “A Mechanistic Model of Three Facets of Meaning,” *Symbols and Embodiment: Debates on Meaning and Cognition*, M.D. Vega, A.M. Glenberg, and A.C. Graesser, eds., Oxford, UK: Oxford University Press, 2008, pp. 195–222.

57. Stanton, C. J., “The Value of Meaning for Autonomous Robots,” *Proceedings of the Tenth International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*, B. Johansson, E. Sahin, and C. Balkenius, eds., Lund University: 2010, pp. 129–136.

58. Taniguchi, T., Nagai, T., Nakamura, T., Iwahashi, N., Ogata, T., and Asoh, H., “Symbol Emergence in Robotics: A Survey,” *Advanced Robotics*, vol. 30, Jun. 2016, pp. 706–728.

59. Gudwin, R., “The Icon Grounding Problem,” *International Journal of Signs and Semiotic Systems*, vol. 1, Mar. 2011, pp. 73–74.

60. Parisi, G. I., Tani, J., Weber, C., and Wermter, S., “Emergence of Multimodal Action Representations from Neural Network Self-Organization,” *Cognitive Systems Research*, vol. 43, Jun. 2017, pp. 208–221.