Comprehensive Introduction to the Notion of Problem Solving Logic for Professional Accountants

By Charles Hoffman, CPA (Charles.Hoffman@me.com)

October 18, 2016 (DRAFT)

Some sort of approach or method is used to solve a problem. Said another way, solving a problem is not generally a "fly by the seat of your pants" or random effort. The best approaches to solving problems are deliberate and methodical.

The way a problem is solved is problem solving logic.

There is a difference between the ways a reasoning engine solves a problem and how a typical software program solves a problem. A reasoning engine is a tuned to solve a specific set of problems. That means that it cannot solve other specific types of problems. A software program can be created to solve any specific problem. That means that to solve each type of problem, new code needs to be written. There are tradeoffs between and "engine" type of an approach and an "ad hoc" or roll-your-own type of an approach. Each approach works, but each has different sets of pros and cons.

Life is full of trade-offs. When evaluating available options the full cost and full benefits of each alternative need to be weighed in order to pick the alternative that best fits your needs.

Fads, misinformation, arbitrary preferences, ignorance, politics, trends, and other such things get in the way of making good decisions[1]. "Knowing one's way about" brings great benefits. The philosopher Nicholas Rescher put it this way[2]:

> ...Knowledge brings great benefits. The release of ignorance is foremost among them. We have evolved within nature into the ecological niche of an intelligent being. In consequence, the need for understanding, for "knowing one's way about," is one of the most fundamental demands of the human condition.

This document helps you know your way about and helps you work through the important area of problem solving logic or problem solving method. Why is this important? More and more information is becoming digital. For example, XBRL-based digital financial reports[3].

To remain relevant in the digital age, professional accountants need to understand how computers solve problems.

---

[1] John F. Sowa, *Fads, Misinformation, Trends, Politics, Arbitrary Preferences, and Standards*, http://xbrl.squarespace.com/journal/2016/9/23/fads-misinformation-trends-politics-arbitrary-preferences-an.html

[2] Wikipedia, *Nicholas Rescher*, retrieved October 18, 2016, https://en.wikipedia.org/wiki/Nicholas_Rescher

[3] *Conceptual Overview of an XBRL-based, Structured Digital Financial Report*, http://xbrlsite.azurewebsites.net/2016/Library/ConceptualOverviewOfDigitalFinancialReporting.pdf

1. Deconstructing the Notion of Problem Solving Logic

In an interview with *Wired* magazine[4], Barak Obama (yes, the president of the United States discussing artificial intelligence) made the following statement about self-driving cars:

> "There are gonna be a bunch of choices that you have to make, the classic problem being: If the car is driving, you can swerve to avoid hitting a pedestrian, but then you might hit a wall and kill yourself. It's a moral decision, and who's setting up those rules?"

This example which relates to self-driving cars points out two things that accounting professionals need to consider when thinking about XBRL-based digital financial reports: (1) who writes the rules, the logic, which software follows, (2) how do you write those rules and put them into machine readable form?

Computers work using the rules of mathematics. Mathematics works using the rules of logic. A problem solving logic is how a computer reasons.

To understand the notion of problem solving logic one first needs to understand the notion of logic and how logic can be applied to solving a problem. This section is dedicated to setting your perspective. The section provides specific definitions, deconstructing the pieces so that we can subsequently put the pieces back together.

1.1. Definition of a reasoning system

Wikipedia defines a reasoning system[5] as "a software system that generates conclusions from available knowledge using logical techniques such as deduction and induction".

The fact is, all computer systems are reasoning systems in that they all automate some type of logic or decision. For example, computing your annual income based on the number of hours worked and the pay rate per hour is reasoning. However, we want to talk about complete reasoning systems such as semantic reasoners or simply reasoner. Here is one definition of a semantic reasoner[6]:

> "A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine, by providing a richer set of mechanisms to work with. The inference rules are commonly specified by means of an ontology language, and often a description language. Many reasoners use first-order predicate logic to perform reasoning; inference commonly proceeds by forward chaining and backward chaining."

[4] Wired, *Barack Obama, Neural Nets, Self-driving Cars, and the Future of the World*, https://www.wired.com/2016/10/president-obama-mit-joi-ito-interview/

[5] Wikipedia, *Reasoning system*, retrieved October 18, 2016, https://en.wikipedia.org/wiki/Reasoning_system

[6] *Semantic Reasoner*, http://hellosemanticweb.blogspot.com/2011/04/semantic-reasoners.html#axzz2URUuQy00

1.2. Definition of a logic

Merriam-Webster provides this simple definition of logic[7]:

Simple definition of logic:

- a proper or reasonable way of thinking about or understanding something
- a particular way of thinking about something
- the science that studies the formal processes used in thinking and reasoning

A logic is simply a set of rules and processes used to reason. Formal logic has been around since about 384 B.C. and was said to have been invented by Aristotle[8]. Logic is a discipline of philosophy. Logic is the study of correct reasoning.

1.3. Definition of a theory

A ***theory***[9] is a prescriptive or normative statement which makes up a body of knowledge about what ought to be. A theory provides goals, norms, and standards. To theorize is to develop this body of knowledge.

A theory is a tool for understanding, explaining, and making predictions about a system. A theory describes absolutes. A theory describes the principles by which a system operates. A theory can be right or a theory can be wrong; but a theory has one intent: to discover the essence of some system.

A theory is consistent if its theorems will never contradict each other. Inconsistent theories cannot have any model, as the same statement cannot be true and false on the same system. But a consistent theory forms a conceptual model which one can use to understand or describe the system. A conceptual model or framework helps to make conceptual distinctions and organize ideas.

Theories are the real thing. A theory describes the object of its focus. A theory does not simplify. Theories are irreducible, the foundation on which new metaphors can be built. A successful theory can become a fact.

Axioms describe self-evident logical principles that no one would argue with. Axioms deal with primitives and fundamentals. An axiom is a premise so evident that it is accepted as true without controversy. ***Theorems*** are deductions which can be proven by constructing a chain of reasoning by applying axioms in the form of IF…THEN statements. A theorem is a statement that has been proven on the basis of previously established theorems or generally accepted axioms.

A ***proof***[10], or formal proof[11], is a set of axioms and theorems that are used to determine if a *theory* is true.

The rules of logic are used to prove a theory[12]. Logic is sequences of reasoning for determining whether a set of axioms and theorems that form some theory are true or false.

---

[7] Merriam-Webster, *Logic definition*, retrieved October 18, 2016, http://www.merriam-webster.com/dictionary/logic

[8] Wikipedia, *Aristotle*, retrieved October 18, 2016, https://en.wikipedia.org/wiki/Aristotle

[9] Wikipedia, *Theory*, retrieved August 29, 2016, https://en.wikipedia.org/wiki/Theory

[10] Richard Hammack, *Book of Proof*, http://www.people.vcu.edu/~rhammack/BookOfProof/

[11] Wikipedia, *Formal Proof*, retrieved October 18, 2016, https://en.wikipedia.org/wiki/Formal_proof

1.4. Putting logic into machine readable form

Description logics[13] are a family of formal knowledge representation logical languages. Description logics have varying levels of expressive power.

One important description logic is *SROIQ*[14]. The *SROIQ* description logic is the basis for the web ontology language, OWL 2 DL[15]. OWL 2 DL was designed be implemented using software. Software can read OWL 2 DL and reason because of the knowledge represented in that machine-readable format. There are many other machine-readable ways of representing such information.

The expressiveness of OWL 2 DL and *SROIQ* description logic is limited; for example, mathematical relations cannot be expressed using that logical language. The point is, while such logics can be put into machine-readable form, not all logics are equivalent and not all problem solving logics are equivalent.

In the next section we will look at several different and powerful problem solving logics.

[12] YouTube, *Crash Course in Formal Logic Part 1*, https://www.youtube.com/watch?v=ywKZgjpMBUU

[13] Wikipedia, *Description Logic*, retrieved October 18, 2016, https://en.wikipedia.org/wiki/Description_logic

[14] Ian Horrocks and Oliver Kutz and Ulrike Sattler, *The Even More Irresistible SROIQ*, http://www.cs.ox.ac.uk/people/ian.horrocks/Publications/download/2006/HoKS06a.pdf

[15] W3C, *OWL 2 Web Ontology Language Primer (Second Edition)*, http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

2. Understanding the Notion of Problem Solving Logic

Creating a problem solving logic is a balancing act. You want the logic to have the maximum in terms of expressiveness. But you want the logic to be safely implementable in software application so that logical catastrophes do not occur which cause systems to crash or provide results that are not reliable or predictable.

2.1. Describing systems formally

Deliberate, rigorous, conscious, skillful execution is preferable to haphazard, negligent, unconscious, inept execution if you want to be sure something works. Engineering a system to make sure it works as designed is a very good thing. Knowledge engineering is the process of representing information in machine-readable form[16].

A digital financial report[17] is a type of formal system. A digital financial report is mechanical and those mechanical aspects of how such a report works can be described using a conceptual model. The *Financial Report Semantics and Dynamics Theory*[18] describes the conceptual model of a digital financial report.

A system such as the digital financial report needs to be described precisely so that professional accountants understand the mechanics of how the system works so that the system can be used effectively and so the system works how the system was intended to work.

Z Notation[19] is an ISO/IEC standard for describing systems precisely. Z Notation is used to describe safety-critical systems such as nuclear power plants, railway signaling systems, and medical devices. But while Z Notation is precise, Z Notation is not machine-readable.

Common Logic[20] (CL), also an ISO/IEC standard, is a framework for a family of logic languages, based on first-order logic, intended to facilitate the exchange and transmission of knowledge in computer-based systems. Common Logic is machine-readable. Further, the logic allowed to be expressed by Common Logic is consciously limited to avoid logical catastrophes[21] which cause systems to break.

Common Logic is about being practical, something business professionals generally tend to like. Common logic is a conscious compromise in order to achieve reliability, predictability, and safety. Common Logic is a "sweet spot" that achieves high expressivity but consciously gives up certain specific things that lead to catastrophic results that cause systems to potentially break making a system unsound; so that a system will be sound. Common Logic

---

[16] *Comprehensive Introduction to Knowledge Engineering Basics for Professional Accountants*, http://xbrlsite.azurewebsites.net/2016/Library/ComprehensiveIntroductionToKnowledgeEngineeringForProfessionalAccountants.pdf

[17] *Conceptual Overview of an XBRL-based, Structured Digital Financial Report*, http://xbrlsite.azurewebsites.net/2016/Library/ConceptualOverviewOfDigitalFinancialReporting.pdf

[18] *Financial Report Semantics and Dynamics Theory*, http://xbrl.squarespace.com/fin-report-sem-dyn-theory/

[19] *Understanding the Importance of Z Notation*, http://xbrl.squarespace.com/journal/2015/9/4/understanding-the-importance-of-z-notation.html

[20] *Understanding Common Logic*, http://xbrl.squarespace.com/journal/2016/6/23/understanding-common-logic.html

[21] *Brainstorming the Idea of Logical Catastrophes or Failure Points*, http://xbrl.squarespace.com/journal/2015/7/25/brainstorming-idea-of-logical-catastrophes-or-failure-points.html

establishes well-thought-out boundaries, allowing creators of systems to "stay within the lines" and if you do, you get a maximum amount of expressiveness with the minimum risk of catastrophic system failure. Thus, you get a more reliable, dependable system.

Semantics of Business Vocabulary and Business Rules[22] (SBVR) is an OMG standard that was designed and built to be logically equivalent to Common Logic.

Rulelog[23] is a logic that is consciously engineered to be consistent with ISO/IEC Common Logic and OMG Semantics of Business Vocabulary and Business Rules. Rulelog is a dialect of W3C's RIF[24]. RuleML[25] is a syntax for implementing rules. Other standard and proprietary syntaxes exist for implementing rules.

What is the point? Ask yourself why ISO/IEC and OMG would go through the trouble to create specifications such as Z Notation, Common Logic, and Semantics of Business Vocabulary and Business Rules? The answer to that question is to enable systems to be described precisely so that they can be implemented successfully using computer software.

Logics can be used to describe systems. Standard logics, such as Common Logic and Semantics of Business Vocabulary and Business Rules enable interoperability. As John F. Sowa put it in *Fads and Fallacies about Logic*[26]:

> "In summary, logic can be used with commercial systems by people who have no formal training in logic. The fads and fallacies that block such use are the disdain by logicians for readable notations, the fear of logic by nonlogicians, and the lack of any coherent policy for integrating all development tools. The logic-based languages of the Semantic Web are useful, but they are not integrated with the SQL language of relational databases, the UML diagrams for software design and development, or the legacy systems that will not disappear for many decades to come. A better integration is possible with tools based on logic at the core, diagrams and controlled natural languages at the human interfaces, and compiler technology for mapping logic to both new and legacy software."

The bottom line is that the best balance between expressive power and safe implementation has been achieved by the ISO/IEC global standard Common Logic. **Common Logic**[27] is a framework for a family of logic languages, based on first-order logic, intended to facilitate the exchange and transmission of knowledge in computer-based systems. That safely expressive sweet spot is also used by the OMG standard **Semantics of Business Vocabulary and Business Rules**[28] which was consciously designed to be logically equivalent to ISO/IEC Common Logic.

The most important thing to realize is that there is a good, safe target in terms of an expressive logic that is also safely implementable in software so catastrophic failures are avoided. Another very good thing is that business professionals don't need to understand the underlying technical details of these logic standards, nor will they every have to deal

---

[22] OMG, *Semantics of Business Vocabulary and Business Rules (SBVR)*, section 2.5 Conformance of an SBVR Processor, page 7, http://www.omg.org/spec/SBVR/1.0/

[23] *Rulelog*, http://ruleml.org/rif/rulelog/spec/Rulelog.html

[24] W3C, *RIF Overview (Second Addition)*, http://www.w3.org/TR/rif-overview/

[25] *RuleML*, http://wiki.ruleml.org/index.php/RuleML_Home

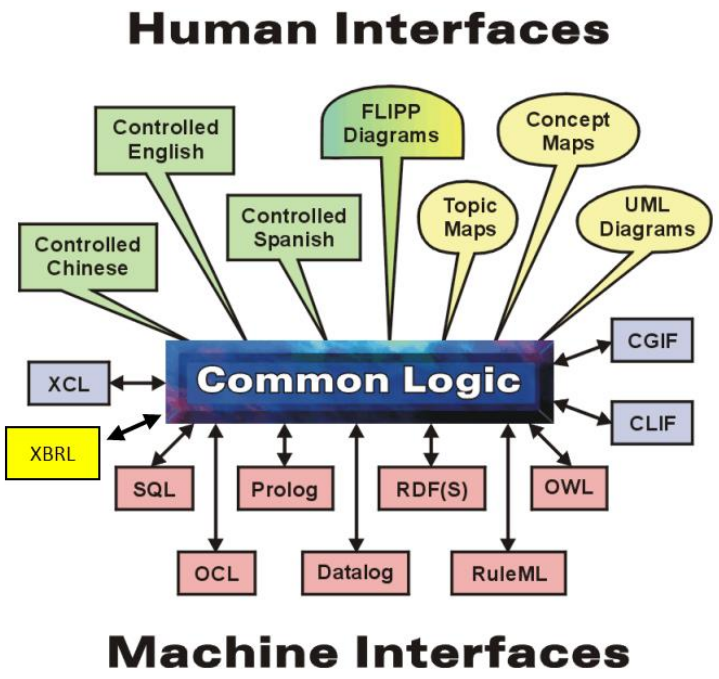
[26] John F. Sowa, *Fads and Fallacies about Logic*, page 6, http://www.jfsowa.com/pubs/fflogic.pdf

[27] *Understanding Common Logic*, http://xbrl.squarespace.com/journal/2016/6/23/understanding-common-logic.html

[28] OMG, *Semantics of Business Vocabulary and Business Rules (SBVR)*, section 2.5 Conformance of an SBVR Processor, page 7, http://www.omg.org/spec/SBVR/1.0/

with them. Higher level languages that follow the foundations set by Common Logic, Semantics of Business Vocabulary and Business Rules, and Rulelog.

The following graphic shows the role Common Logic[29] plays, establishing a family of logical dialects shared between different software syntax implementations: (note that this graphic was modified, XBRL was added)



The language in which a problem is stated has no effect on complexity. Reducing the expressive power of a logic does not solve any problems faster; its only effect is to make some problems impossible to state[30].

2.2. XBRL is a problem solving logic that should be equivalent to Common Logic, SBVR, and RuleLog

The XBRL technical syntax is a global standard logic for representing knowledge. While much of the logic such as XBRL elements, relations between elements, mathematical relations between concepts and facts (XBRL calculation relations and XBRL Formula relations), dimensional relationships between concepts and facts, and other such relations (expressible using XBRL definition relations); not all such relation logic is standard.

XBRL Formula processers have specific deficiencies in their processing capabilities[31]. To overcome these deficiencies, the following capabilities must exist or need to be added to XBRL Formula Processors:

[29] John F. Sowa, *Common Logic: A Framework for a Family Of Logic-Based Languages*, page 5, http://www.jfsowa.com/ikl/SowaST08.pdf

[30] John F. Sowa, *Fads and Fallacies about Logic*, page 5, http://www.jfsowa.com/pubs/fflogic.pdf

- Support **normal global standard functionality** that high-quality XBRL Formula processors support (i.e. Arelle, UBmatrix/RR Donnelley, Fujitsu, Reporting Standards, etc.)
- **Support inference** (i.e. deriving new facts from existing facts using logic, what inference engines do)
- Improved support validation and use of **structural relations** (i.e. XBRL Taxonomy functions; this was consciously left out of the XBRL Formula specification in order to focus on XBRL instance functionality)
- Support **forward chaining** and possibly also backward chaining in the future (i.e. chaining was also proposed but was left out of the XBRL Formula specification)
- Support a **maximum amount of Rulelog logic** which is safely implementable and is consistent with ISO/IEC Common Logic and OMG Semantics of Business Vocabulary and Business Rules
- **Additional XBRL definition arcroles** that are necessary to articulate the Rulelog logic, preferably these XBRL definition relation arcroles would end up in the XBRL International Link Role Registry and be supported consistently by all XBRL Formula processors (i.e. these general arcroles, and these financial disclosure related arcroles; this human readable information is helpful to understand the arcroles)

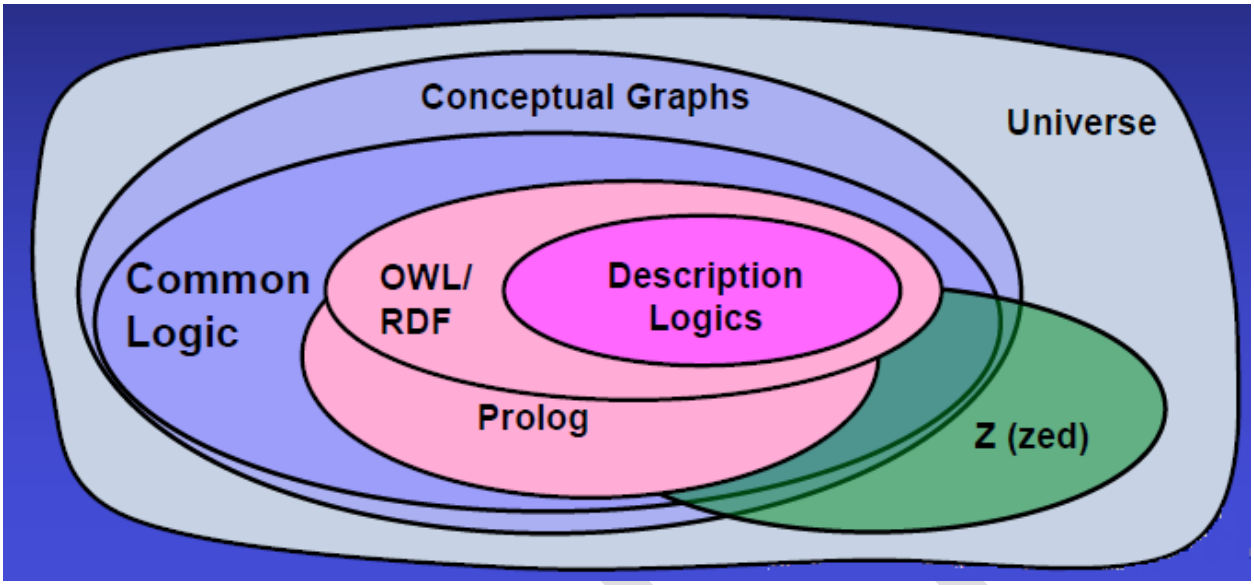
While added functionality might not be global standard functionality, the functionality is necessary to prove the logic of US GAAP based financial reporting or IFRS based financial reporting. US GAAP and IFRS semantics are relatively clear. What is not clear to some business professionals is how to represent that meaning using the XBRL global standard. Proprietary techniques for applying XBRL can be used to fill any gap. However, the logical rules used by any proprietary techniques should follow the logic of Common Logic, SBVR, and RuleLog.

2.3. Comparing expressiveness

Expressiveness is the set of things that can possibly be expressed by some language. Below is a graphic which shows the relative expressiveness of Common Logic and Z Notation relative to the universe of all possible expressiveness[32].

[31] Specific Deficiencies in Capabilities of Existing XBRL Formula Processors, http://xbrl.squarespace.com/journal/2016/9/26/specific-deficiencies-in-capabilities-of-existing-xbrl-formu.html

[32] *Common Logic in Support of Metadata and Ontologies*, Page 2, Retrieved June 24, 2016, http://cl.tamu.edu/docs/cl/Berlin_OpenForum_Delugach.pdf



Not even included in this comparison, because the expressiveness is so low is the Comma Separated Values[33] (CSV) technical format. CSV is a very popular data format and it is very easy to use. But CSV does nothing to help assure the quality of data represented in this technical format. Basically, you can articulate a simple list in CSV and you cannot provide information which helps a user of the information understand that the information is consistent with expectations in terms of representation (i.e. quality is high).

2.4. Understanding the relation between expressiveness and reasoning capacity

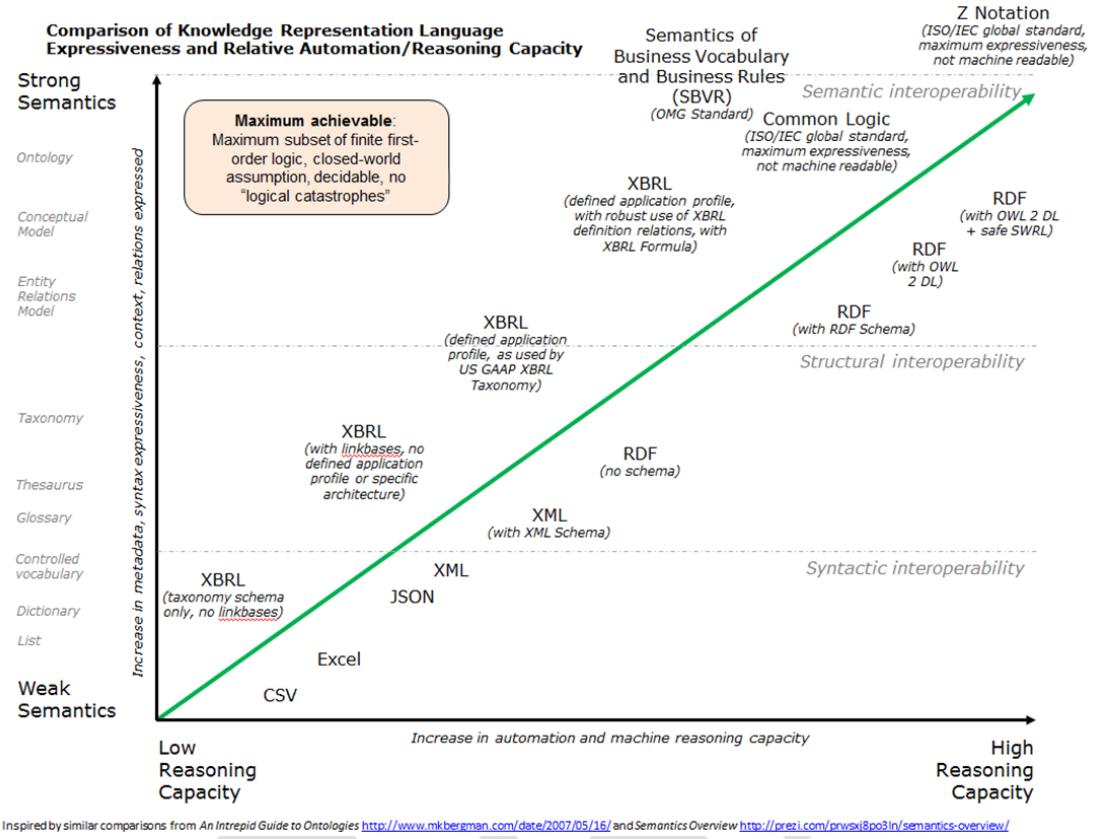
Why is the expressiveness of a language important? There are two reasons. First, the more expressive a language the more that language can provide in terms of describing the information being represented and verifying the consistency of what is being represented with expectations (i.e. quality).

But secondly, the more expressive the language is; the more a computer can do for a user of an application in terms of reasoning capacity. The higher the expressiveness, the better the problem solving logic. So, the two work together. Both the quality of the information being processed is higher and what the software can do is higher because of both the expressiveness of the language but also because of the quality of the information which is represented.

Another way to say this is "nonsense in, nonsense out". As has been pointed out, the only way to have a meaningful exchange of information is the prior existence of technical syntax rules (the language syntax), business domain semantics (the descriptive and structural metadata), and the workflow rules (protocols for what to do if say an amended financial report is submitted to a regulator).

This graphic below compares the relative knowledge representation language expressiveness and the relative automation and reasoning capacity which is achievable using that language.

[33] Wikipedia, *Comma Separated Values*, retrieved August 28, 2016, https://en.wikipedia.org/wiki/Comma-separated_values



Inspired by similar comparisons from *An Intrepid Guide to Ontologies* http://www.mkbergman.com/date/2007/05/16/ and *Semantics Overview* http://prezi.com/prwsxj8po3ln/semantics-overview/

At the bottom left hand corner of the graphic you see "CSV" which is not expressive (i.e. weak semantics). At the top left you see the ISO/IEC standard "Z Notation" which is highly expressive (i.e. strong semantics). But remember, Z Notation is not machine-readable. But you also see Common Logic, Semantics of Business Vocabulary Rules, and XBRL as having strong semantics. Those three formats are all machine-readable.

No knowledge representation language is 100% complete. Each has specific, knowable limitations. One must be conscious of such limitations when creating a representation of some problem domain in machine readable form.

A representation language or framework which cannot be measured for simplicity is a recipe for unnecessary complexity. Conscientious knowledge engineers are compelled to express a problem domain's conceptual model as richly as possible. With a highly-expressive language at a knowledge engineer's disposal it is possible to think through different representational options at a level of detail that is impossible with a weaker-expressive language. Stronger languages push one more than one using a weaker language. Testing pushes one more than not using testing toward greater accuracy and comprehensiveness. As is said, "Ignorance is bliss." Limitations of expressivity of the representation language used should be exposed so that the limitations become conscious.

2.5. Specific expressiveness features comparison

The following is a comparison of specific features of expressiveness[34]:

KRR Features Comparison: Rulelog Shines

System / Feature	Rulelog Rules - e.g., Ergo	Datalog Rules - e.g., Jena, SWRL, Ontobroker, SPIN	Production Rules - e.g., IBM, Oracle, Red Hat	Prolog - e.g., SICStus, SWI, XSB	FOL & OWL-DL - e.g., Vampire, Pellet, Prover9	ASP Solvers - e.g., DLV, CLASP
Semantic & on standardization path	✓	✓	restricted case	restricted case	✓	✓
Basic expressiveness						
• Datalog LP	✓	✓	✓	✓	✓	✓
• Logical functions	✓	×	×	✓	✓	✓
• General formulas	✓	×	×	×	✓	×
Full Meta expressiveness						
• Higher-order syntax, provenance	✓	×	×	×	×	×
• Defeasibility & well founded negation	✓	×	×	×	×	×
• Restraint bounded rationality	✓	×	×	×	×	×
• Probabilistic	✓	×	×	×	×	×
Efficiency						
• Goal-directed	✓	× (except Jena)	×	✓	✓	✓
• Full LP tabling with dependency-aware updating	✓	×	×	× (except XSB)	×	×
• Polynomial time complexity	✓	✓	✓	×	×	×

16

Note that it is unknown if any of these knowledge representation logics supports a multidimensional model (i.e. without the user having to create that model). KRR – Knowledge representation and reasoning.

2.6. Understanding why logical catastrophes break systems

A logical catastrophe is a failure point. Logical catastrophes must be eliminated. Systems should never have these failure points. A basic example of a catastrophic failure is creating metadata that puts a process into an infinite loop that the software will not recover from. This type of catastrophic failure is resolved by simply not allowing the conceptual model to include such structures which cause the possibility of infinite loops. It really is that straight forward.

Here are other types of logical catastrophes:

[34] Coherent Knowledge, *KRR Features Comparison*, http://coherentknowledge.com/wp-content/uploads/2013/05/talk-main-v14-post.pdf#page=16

- **Undecidability**: If a question cannot be resolved to a TRUE or FALSE answer; for example if the computer returns UNKNOWN then unpredictable results can be returned. Logic used by a computer must be decidable. Saying this another way, three-value logic[35] (i.e. TRUE, FALSE, and UNKNOWN are all valid) is a valid for of logic. However, if some people use two value logic (TRUE, FALSE) and others use three-value logic, big problems can occur.
- **Infinite loops**: If a computer somehow enters an infinite loop from which it cannot return because of a logic error or because the logic is too complex for the machine to work with; the machine will simply stop working or return nonsense.
- **Unbounded system structures or pieces**: Systems need boundaries for them to work correctly. Boundaries must be well defined so that they are well understood. If a system does not have the proper boundaries, then a machine can become confused or not understand how to work with information that is provided. For example, if an entirely new class of concept is added to a system that the system has no knowledge of, the system will not understand how to process that class of concept and will fail.
- **Unspecific or imprecise logic**: Confusing precise results with the capabilities of a computer to provide a statistically created result can cause problems. It is not expected that the business system at the level of describing the things in the system be able to support "fuzzy logic" or "probabilistic reasoning" or other such functionality.

2.7. Understanding the critical importance of decidability

There are two fundamental approaches to viewing a system that one could take: the open world assumption (i.e. two-value logic) and the closed world assumption (i.e. three-value logic). Formal logic and relational databases use the closed world assumption. Decidability means that a conclusion can be reached.

- In the **open world assumption** a logical statement cannot be assumed true on the basis of a failure to prove the logical statement. On a World Wide Web scale this is a useful assumption; however a consequence of this is that an inability to reach a conclusion (i.e. not decidable).
- In the **closed world assumption** the opposite stance is taken: a logical statement is true when its negation cannot be proven; a consequence of this is that it is always decidable. In other applications this is the most appropriate approach.

So each type of system can choose to make the open world assumption or the closed world assumption based on its needs. Because it is important that a conclusion as to the correct mechanics of a financial report is required because consistent and correct mechanics are necessary to making effective use of the information contained within a financial report; the system used to process a financial report must make the closed world assumption.

[35] Wikipedia, *Three-valued Logic*, retrieved October 19, 2016, https://en.wikipedia.org/wiki/Three-valued_logic

2.8. Setting the right expectation by understanding the capabilities of computers

First-order logic has limitations[36]. Business professionals need to understand these limitations so that they understand what computers can and cannot do, what is hard and what is easy to implement using computers, and to otherwise set their expectations appropriately. Remember, computers cannot perform magic. Computers fundamentally follow the rules of mathematics which follow the rules of formal logic. It really is that straight forward.

It is difficult to get computers to effectively work with information such as the following:

- fuzzy expressions: "It **often** rains in autumn."
- non-monotonicity: "Birds fly, penguin is a bird, but a penguin does not fly."
- propositional attitudes: "Eve **thinks** that 2 is not a prime number." (It is true that she thinks it, but what she thinks is not true.)
- modal logic
    - possibility and necessity: "It is **possible** that it will rain today."
    - epistemic modalities: "Eve **knows** that 2 is a prime number."
    - temporal logic: "I am **always** hungry."
    - deontic logic: "You **must** do this."

While it is possible to implement this sort of functionality within computer systems using technologies such as probabilistic reasoning[37], those systems will be less reliable and significantly more difficult to create. On the other hand, probabilistic reasoning can provide value. The bottom line is this: what are the boundaries of the system?

2.9. Procedural versus declarative rules

Problem solving logic is expressed in the form of logical business rules[38].

The *Business Rules Manifesto*, Article 4[39], points out that business rules should be *declarative* rather than *procedural*. However, either the declarative or procedural approach will work, which approach you use can be an arbitrary preference[40]. However each approach does have pros and cons.

The declarative approach has important advantages including that your business rules become reusable across both processes and software platforms. As such, the rules become both highly re-engineerable and highly re-deployable.

---

[36] Martin Kuba, Institute of Computer Science, *OWL 2 and SWRL Tutorial*, *Limitations of First-order logic expressiveness*, http://dior.ics.muni.cz/~makub/owl/

[37] Wikipedia, *Probabilistic Logic*, retrieved August 28, 2016, https://en.wikipedia.org/wiki/Probabilistic_logic

[38] *Comprehensive Introduction to Business Rules for Professional Accountants*, http://xbrlsite.azurewebsites.net/2016/Library/ComprehensiveIntroductionToBusinessRulesForProfessionalAccountants.pdf

[39] *Business Rules Manifesto, Article 4. Declarative, Not Procedural*, http://www.businessrulesgroup.org/brmanifesto.htm

[40] John F. Sowa, *Fads, Misinformation, Trends, Politics, Arbitrary Preferences, and Standards*, http://xbrl.squarespace.com/journal/2016/9/23/fads-misinformation-trends-politics-arbitrary-preferences-an.html

Declarative involves stating THAT something is the case. Procedural involves stating HOW to do something.

The following is a simple example of procedural rules and declarative: Suppose you desire a cup of coffee.

Procedural:

1. Go to kitchen.
2. Get water, coffee, sugar, cream.
3. Heat the water on the stove until the water boils.
4. Put the coffee, sugar, and cream into the water.
5. Bring the result to me.

Declarative:

1. Get me a cup of coffee.

Taking a procedural approach you define the entire process and provide each step necessary to obtain the desired result. Taking a declarative approach you state the desired result, and let the system determine the best way to get that result; all you care about is the result without worrying how the result will be achieved.

A procedure is used in only one way, but a declarative specification can be used in many different ways[41].

Business rules should not be mixed within software application code. Why? Three reasons. First, if business rules are within application code then it takes a programmer to change the code. Second, if the business rules are embedded within one software application that it is challenging to reuse those same rules within another application. Third, sharing business rules becomes easy.

2.10. General versus specific problem solving logics

Problem solving logics can be general or specific. Another term used for general is "weak (basic) problem-solving method. Another term for specific is "strong problem-solving method". Both the general and specific logics have advantages and disadvantages.

General problem solving logics are widely applicable to many problem domains which is an advantage. However, with this flexibility comes the price of a harder to use problem solving logic.

Specific problem solving logics are limited and generally applicable to one specific problem domain. This limitation to one problem domain can be seen as a disadvantage. However, an advantage of the specific nature of the problem solving logic is that it tends to be easier to use because it is specific to the problem domain.

XBRL tends to be limited to business reporting and financial reporting.

[41] John F. Sowa, *Fads and Fallacies about Logic*, page 3, http://www.jfsowa.com/pubs/fflogic.pdf

2.11. Approaches for representing information logically in machine readable form

There are three generally used approaches to representing information logically in machine-readable form:

- **Natural language format**: A natural language which is parsed.
- **Truth table-type format**: A table-type or "truth table" type format.
- **Graphical format**: A graphical format.

2.12. Functional layers or categories of problem solving logic

Problem solving logics can be grouped into "functional layers" or "functional categories" or "functional groups"[42]:

- **Sequence, process or flow:**
  - **procedural logic** – model sequence, loop, or iterative procedures
  - **flow logic** – fully automated sequence of operations, actions, tasks, decisions, rules.
  - **workflow logic** – type of flow logic, semi-automated or manual processes that need an action to be taken from outside the system by another system or human.
- **Information compliance, quality, consistency, completeness, accuracy**:
  - **validation logic**: validate action assertions.
  - **decision logic**: type of validation logic, handles execution que and conflict resolution.
  - **inference logic**: deviations which derives new facts using existing facts, rules, and logical or mathematical reasoning.
  - **structural relations logic**: enforces structural relationships within a representation model

These categories or layers can be useful in grasping the potential power of a problem solving logic.

2.13. Details of problem solving logic features

The comparison below is a DRAFT of a detailing of problem solving logic features that are included in ISO/IEC standard Common Logic. It also tries to articulate the functionality offered by software products to meet the problem solving logic needs when working with a digital financial report.

(NOTE that this is a work in progress.)

[42] *Logic*, http://wiki.flexrule.com/index.php?title=Logic

Problem Solving Logic Feature	ISO/IEC Common Logic, OMG SBVR, RuleLog	OWL 2 DL , SROIQ Description Logic	FlexRule	Ergo Suite	XBRL Formula Processor
Basic expressiveness (relational database logic (DATALOG)					
Set theory (cardinality, association, aggregation)					
Constraints and data types					
Referential integrity					
Definition of Terms					
Temporal reasoning					
Assertions					
Structural assertions (including mereology or part-whole)					
Action assertions					
Mathematical assertions (add, subtract, multiply, divide)		*using RIF*			
Derivations/inference logic (deriving facts)					
Mathematical inference					
Logical inference					
Flow logic (procedural, workflow)					
Procedural					
Workflow					
Chaining					
Forward chaining					
Backward chaining (goal directed)					
Other logical functionality					
Fiscal period reasoning (inherent understanding of)					
Multidimensional model (inherent understanding of)		*using DataCube*			
Object oriented representations (frames, see CLIPS)					
Logical functions					
Full meta expressiveness					
Higher-order syntax, provenance					
Defeasibility and well-founded negation					
Restraint bounded rationality					
Probabilistic reasoning					