1. Comprehensive Introduction to Problem Solving Logic

Some sort of approach or procedure 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. 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.

1.1. Deconstructing the Notion of Problem Solving Logic

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

---

[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

[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/

> "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.

1.1.1.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."

1.1.2.Definition of a logic

Logic is a formal system for enabling precise communication. A logic[7] is a set of principles underlying the arrangement of elements so as to perform some task. Merriam-Webster provides this simple definition of logic[8]:

- 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

---

[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

[7] Richard Hammack, Virginia Commonwealth University, *Book of Proof*, http://www.people.vcu.edu/~rhammack/BookOfProof/BookOfProof.pdf

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

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[9]. Logic is a discipline of philosophy. Logic is the study of correct reasoning.

The purpose of a logic is to communicate about some topic. Logic can be used to deduce complex principles from a commonly understood and agreed upon set of basic assumptions.

The same principles of logic work in all sciences and business domains, these logical principles are universal. One starts with a clearly stated and generally accepted set of hypotheses and perhaps some previously proven principles called theorems[10]. Each of these hypotheses and theorems state that if some situation occurs, then some other situation must also occur. Professional accountants understand this as IF…THEN statements.

1.1.3.Definition of a theory

A ***theory***[11] 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 a 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 in 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***[12], or formal proof[13], is a set of axioms and theorems that are used to determine if a *theory* is true.

---

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

[10] Wikipedia, *Theorem*, retrieved June 20, 2017, https://en.wikipedia.org/wiki/Theorem

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

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

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

The rules of logic are used to prove a theory[14]. Logic is sequences of reasoning for determining whether a set of axioms and theorems that form some theory are true or false.

1.1.4.Putting logic into machine readable form

Description logics[15] are a family of formal knowledge representation logical languages. Description logics have varying levels of expressive power.

One important description logic is *SROIQ*[16]. The *SROIQ* description logic is the basis for the web ontology language, OWL 2 DL[17]. OWL 2 DL was designed to 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.

1.2. Fundamentals of logic

Professional accountants use logic informally every day. For example, something that you probably learned in school is BASE; beginning balance + additions – subtractions = ending balance. If you know any three facts, you can always find the forth fact.

Here is an example of logic. Suppose you were trying to find the subtractions from some account.

1. You know that the beginning balance of the account is $5,000
2. You know that the additions to the account was $1,000
3. You know the ending balance of the account is $2,000
4. You know that beginning balance + additions – subtractions = ending balance
5. You can derive the fact that the subtractions to the account are $4,000 using the information provided in #1, #2, #3, and #4 and using the rules of logic

In this example you are using logic to reliably derive the subtractions from some account even though you do not know the actual value of the subtractions. And in doing so you are adding new information to your base of knowledge.

---

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

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

[16] 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

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

There is an important point to recognize here. Logic is the process of deducing information correctly; logic is not about deducing correct information. Consider the example above again. Suppose you were give incorrect information for the beginning balance of the account you are analyzing, that the beginning balance was $9,000 rather than $5,000 as stated above.

Our deduction that the subtractions are $4,000 is now untrue. But the logic is perfectly correct; the information was pieced together correctly, even if some of that information was false.

Understanding the distinction between correct logic and correct information is important because it is important to follow the consequences of an incorrect assumption.

Ideally, we want both our logic to be correct and the facts we are applying the logic to, to be correct. But the point here is that correct logic and correct information are two different things.

If our logic is correct, then anything we deduce from such information will also be correct.

1.2.1.Logical systems

A logical system[18] is an organization of terms and rules used for the analysis of information deduction. It consists of a language which is used to construct sentences and rules for deriving sentences.

Important properties that logical systems are:

- **Consistency** which means that statements or facts don't contradict one another.
- **Validity** which means that the system's rules never allow a false inference of a statement or fact from true premises.
- **Completeness** which means that if a rule is true, the rule can be proven and therefore the rule is justifiable.
- **Soundness** which means that any information in the form of a statement or fact that is part of the system is true.

1.2.2.Knowledge is justified true belief

Knowledge is a familiarity, awareness, or understanding of someone or something, such as facts, information, descriptions, or skills, which is acquired through *experience* or *education* by perceiving, discovering, or learning[19].

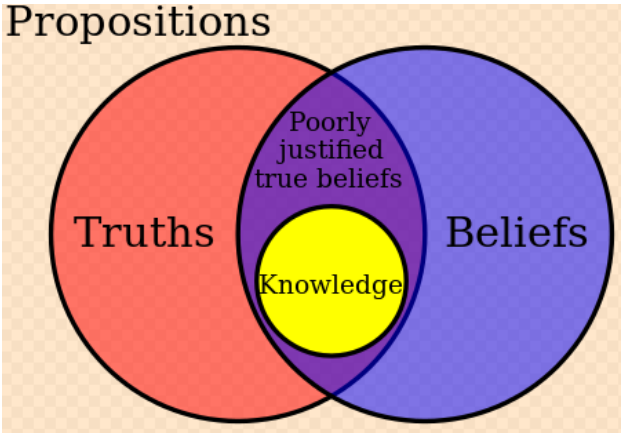
There are three minimum conditions for facts and information to be considered knowledge:

[18] Wikipedia, *Logical Systems*, retrieved June 3, 2017, https://en.wikipedia.org/wiki/Logic#Logical_systems

[19] Wikipedia, *Knowledge*, retrieved June 2, 2017, https://en.wikipedia.org/wiki/Knowledge

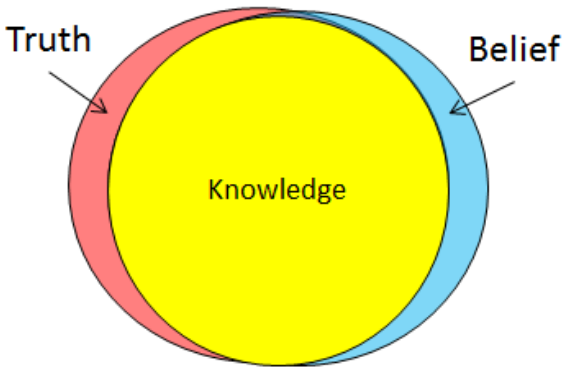
- The information must be **true**.
- You consciously **believe** that the information is true.
- **Justification** is present in the form of sufficient evidence to prove that the information is true. (i.e. a person has sufficient justification for believing what they believe)

The following is a diagram[20] which shows the intersection of what is true, what is believed to be true, and knowledge:



While philosophy likes to debate the meaning of knowledge, and while the majority agrees that knowledge is justified true belief; there are others that do not believe that knowledge is justified true belief. In fact, there is a famous example, the Gettier problem[21], that shows a flaw in that definition of knowledge.

But what if you could create a logical system that has consistency, validity, completeness, and soundness and that the overlap between truth and belief is high providing the maximum amount of knowledge:



[20] Wikipedia, *Intersection of Knowledge*, retrieved June 2, 2017, https://en.wikipedia.org/wiki/Belief#/media/File:Classical_definition_of_Kno.svg
[21] Wikipedia, *Gettier problem*, retrieved June 2, 2017, https://en.wikipedia.org/wiki/Gettier_problem

Such a system is creatable. Engineering is the application of a systematic, disciplined, quantifiable, methodical rigorous approach to the development, operation, and maintenance of something. Those who are skilled in logic can build such systems and such systems are very useful.

1.2.3.Sets

A set[22] is a well-defined collection of objects. Sets have members. The criteria for membership in a set must be well understood. Three set operations are important to understand:

- **Union**: All objects that are in set "A" plus all the objects in set "B" (i.e. objects that are either in set A or in set B or in both sets A and B).
- **Intersection**: All objects that are common to both sets "A" and "B".
- **Complement**: All objects that are members of set "A" but NOT members of set "B".

The notion and rules of sets can be leveraged by logical systems. For example, relational databases leverage set theory.

1.2.4.Logical statements

In logic, a statement is a sentence that is either true or false. You can think of statements as pieces of information that are either correct or incorrect. And therefore, statements are pieces of information that you apply logic to in order to derive other pieces of information which are also statements.

Here are some examples of statements:

- "Assets = Liabilities and equity", i.e. the accounting equation[23].
- "Beginning balance + additions – subtractions = ending balance", i.e. as stated above and the definition of a roll forward.
- "Originally stated balance + adjustments = Restated balance"
- Some types of revenue are operating.
- Assets for the consolidated entity as of the balance sheet date of December 31, 2016 is $45,000.

A financial report discloses facts. One fact in a report is distinguishable from another fact in the report by the characteristics of each of the two facts. One fact in a report can be related to another fact in a report and the relation between those two facts should be consistent with expectations which are established by rules.

Logic is about the correct methods that can be used to prove that a statement is true or false. To prove that a statement is true, we start with statements other statements that are proven to be true and use logic to deduce more and more complex statements until finally we obtain a statement that we are looking to

[22] Wikipedia, *Set*, retrieved June 20, 2017, https://en.wikipedia.org/wiki/Set_(mathematics)

[23] Wikipedia, *Accounting Equation*, retrieved June 2, 2017, https://en.wikipedia.org/wiki/Accounting_equation

determine if the statement is true or false. Of course some statements are more difficult to prove than others; in this resource we will concentrate on statements that are easier to prove to help you learn the basics of logic. But the point is this: In proving that statements are true, we use logic to help us understand statements and to combine pieces of information to produce new pieces of information.

As your skills grow you can create increasing complex statements.

A financial report is complex logical information[24]. XBRL-based financial reports are machine-readable structured information.

1.2.5. Conditional statements

Another way to represent statements is in the IF…THEN format. These are called conditional statements. Here is an example of a conditional statement[25].

- IF Assets = Liabilities and Equity; THEN the balance sheet balances.

Note that you may not be able to reverse a conditional statement. For example, the above conditional statement reversed would be:

- IF the balance sheet balances; THEN Assets = Liabilities and Equity.

The "IF" part of a conditional statement is called the *hypothesis* and the "THEN" part is called a *conclusion*. There are other terms that are used such as the *antecedent* and *consequent*.

1.2.6. Truth tables

A truth table helps you figure out whether a statement is TRUE or FALSE, generally conditional statements.

Here is an example of a truth table:

Assets Exists	*Liabilities and Equity Exists*	*Assets exists AND Liabilities and Equity Exists*	*NOT (Assets Exists AND Liabilities and Equity Exists)*	*NOT (Assets Exists)*	*NOT (Liabilities and Equity Exists)*	*NOT (Assets exists OR Liabilities and Equity Exists)*
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

[24] Charles Hoffman, *Processing Complex Logical Information or Structured Knowledge*, http://xbrl.squarespace.com/journal/2017/4/22/processing-complex-logical-information-or-structured-knowled.html

[25] Wikipedia, *Material Conditional*, retrieved June 2, 2017, https://en.wikipedia.org/wiki/Material_conditional

1.2.7.Logical equivalence

Logical equivalence[26] means that two statements mean *exactly* the same thing, they are logically equivalent, they have the same logical content.

1.2.8.Logical contradiction

A logical contradiction is a statement that is false.

1.2.9.Logical operators

Logic has the following fundamental low-level logical operators:

- Logical **AND**
- Logical **OR**
- Logical **NOT**
- Logically **equivalent to**
- Logically **NOT equivalent to**

Logical operators are used to combined two statements to form some new statement. These low-level logical operators are used to create higher-level logical operators such as:

- Add (+)
- Subtract (-)
- Multiply (*)
- Divide (/)

And they can also be used to create relational operators:

- Greater than (>)
- Less than (<)
- Equal to (=)

The point here is that the basic low-level logical operators are the foundation of all problem solving logic[27]. There are different syntaxes of logical operators[28], such a Z Notation. And logical operators can be used for many different things such as designing electrical circuits[29]. But all the low-level logical operators are the same.

1.2.10. Types of reasoning

The following is a summary of the approaches to reasoning which can be used:

---

[26] Wikipedia, *Logical Equivalence*, retrieved June 2, 2017, https://en.wikipedia.org/wiki/Logical_equivalence

[27] Charles Hoffman, *Comprehensive Introduction to Problem Solving Logic*, http://xbrlsite.azurewebsites.net/2017/IntelligentDigitalFinancialReporting/Part01_Chapter02.5_ComprehensiveIntroductionToProblemSolvingLogic.pdf

[28] Jonathan Jacky, *Glossary of Z Notation*, https://staff.washington.edu/jon/z/glossary.html#logic

[29] Updated by Richard Bigwood, *Basic Gates and Functions*, http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/

- **Deductive or direct reasoning**: Deductive reasoning is the process of finding a direct chain of reasoning, IF…THEN statements, that explicitly bridge some gap between what you know and your hypothesis.
- **Inductive or indirect reasoning**: Inductive reasoning, also known as reasoning by contradiction or *reductio ad absurdum* (reduced to absurdity), is the process of proving that all known alternatives to your hypothesis are not true and therefore your hypothesis must be true because all other hypothesis are not true and you cannot prove that your hypothesis is false.

Another term for inductive or indirect reasoning is logical inference.

1.2.11. Logical inference

Logical inference is the process of deriving new information from one or more existing pieces of information, deducing a conclusion about that information using the rules of logic. For example,

- Suppose you know that Assets = $2,000
- Suppose you know that Current assets = $500
- Suppose you know that Assets = Current assets + Noncurrent assets

Using the information provided above, you can use the rules of logic to derive the value of Noncurrent assets to be $1,500 because Assets ($2,000) = Current assets ($500) + Noncurrent assets (UNKNOWN); but using the rules of math you solve for the value of the UNKNOWN; Assets ($2,000) – Current Assets ($500) = Noncurrent assets (UNKNOWN); finally you get to Noncurrent assets = $1,500.

You are not guessing. You are using logic to determine, accurately, what the value of Noncurrent Assets is based on other facts that you know.

1.2.12. Final thoughts about logic

It is important to understand why you want to take the time to understand logic. Here are the primary reasons:

- Applying logic correctly helps you understand what something means.
- The rules of inference provide a system which can be used to reliably derive new information from existing information.
- Logic helps you not only understand the meaning of statements; it also helps you reliably produce new meaningful statements.

Logic is the glue that holds sets of statements together and helps you understand the exact, precise meaning of statements. Logic is a common language that can be agreed upon which enables communication.

Logic helps you get to true and unambiguous meaning and helps you reconcile your true and unambiguous meaning to the true and unambiguous meaning of others. Computers are machines that work using the rules of logic.

Logic helps to keep everyone on the same page.

1.2.13. Subjectivity and professional judgement

Accounting and financial reporting allow for subjectivity and professional judgement. This is not inconsistent with logic. Professional judgement allows an accountant to pick between allowed alternatives. Professional judgement allows an accountant to determine if something is material. Professional judgement allows for other sorts of subjectivity. However, there is no professional judgement when it comes to things such as whether a roll up actually rolls up or a roll forward actually rolls forward. Roll ups, roll forward, and many other logical, mechanical, and mathematical relations are truths. These higher-level truths are not open to interpretation and not subject to professional judgement.

Not everything in accounting and financial reporting is subjective. Only some things are. Correctly separating what is subjective from what is objective is critical. There are times where this can be ambiguous because of the rules of US GAAP. US GAAP was not designed to be ambiguous, but it was designed to allow for professional judgement.

1.3. 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.

1.3.1.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[30].

A digital financial report[31] is a type of formal system. Many aspects of a digital financial report are 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*[32] describe 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. There are many tools that can be used to describe a system.

---

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

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

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

Z Notation[33] 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[34] (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[35] 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 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[36] (SBVR) is an OMG standard that was designed and built to be logically equivalent to ISO/IEC Common Logic.

Rulelog[37] 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[38]. RuleML[39] is a syntax for implementing rules.

SHACL[40] (Shapes Constraint Language) is a logic that is consciously engineered to work in a "closed world" similar to a relational database. SHACL is a W3C recommendation. SHACL is a language for validating RDF graphs against a set of conditions. SHACL is used to perform closed-world constraint checks on RDF-based data.

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.

---

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

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

[35] *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

[36] 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/

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

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

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

[40] W3C, *Shapes Constraints Language*, http://www.w3.org/TR/shacl/

Information technology professionals will likely never agree on which specific implementation syntax is best. As such, there will likely be multiple technology stacks. But, the different technology stacks should be logically interoperable[41]. Multiple independent, but logically interoperable, stacks of languages; and the XBRL stack should be an option.

Logics can be used to describe systems. Standard logics, such as Common Logic and Semantics of Business Vocabulary and Business Rules, RuleLog, and SHACL enable interoperability. As John F. Sowa put it in *Fads and Fallacies about Logic*[42]:

> "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[43] 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[44] 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 ever have to deal with them. Higher level languages that follow the foundations set by Common Logic, Semantics of Business Vocabulary and Business Rules, Rulelog, and SHACL.

The following graphic shows the role Common Logic[45] plays, establishing a family of logical dialects shared between different software syntax implementations: (note that this graphic was modified, XBRL was added)

---

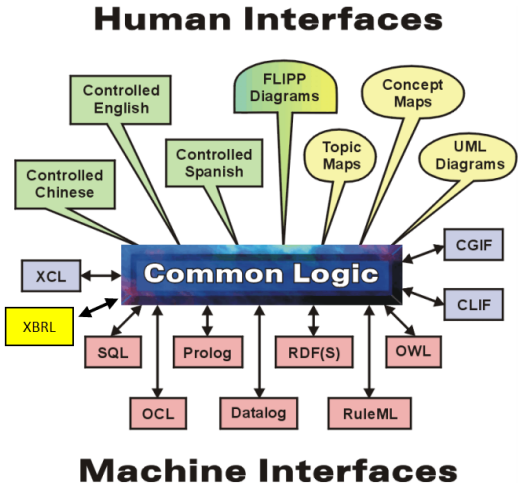
[41] Michael Kifer, Jos de Bruijn, Harold Boley, and Dieter Fensel, *A Realistic Architecture for the Semantic Web*, http://www3.cs.stonybrook.edu/~kifer/TechReports/msa-ruleml05.pdf

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

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

[44] 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/

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



The language or syntax 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[46].

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

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[47]. To overcome these deficiencies, the following capabilities must exist or need to be added to XBRL Formula Processors:

- 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)

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

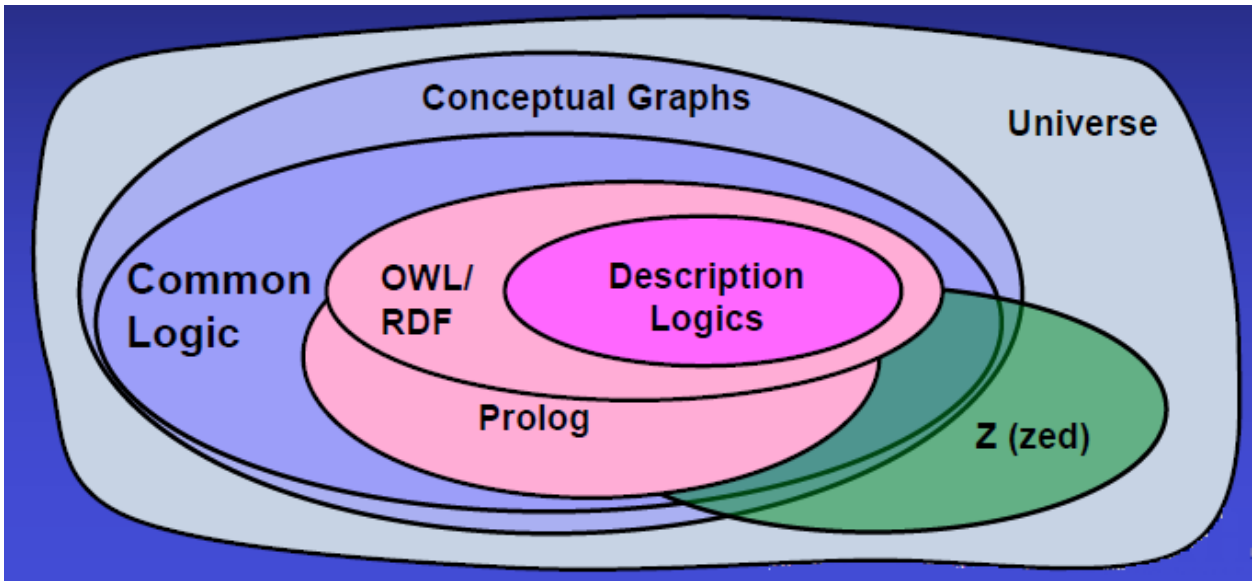
[47] Specific Deficiencies in Capabilities of Existing XBRL Formula Processors, http://xbrl.squarespace.com/journal/2016/9/26/specific-deficiencies-in-capabilities-of-existing-xbrl-formu.html

- 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 convey 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, RuleLog, and SHACL.

1.3.3. 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[48].



Not even included in this comparison, because the expressiveness is so low is the Comma Separated Values[49] (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 in a standard way which helps a user of

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

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

the information understand that the information is consistent with expectations in terms of representation (i.e. quality is high).

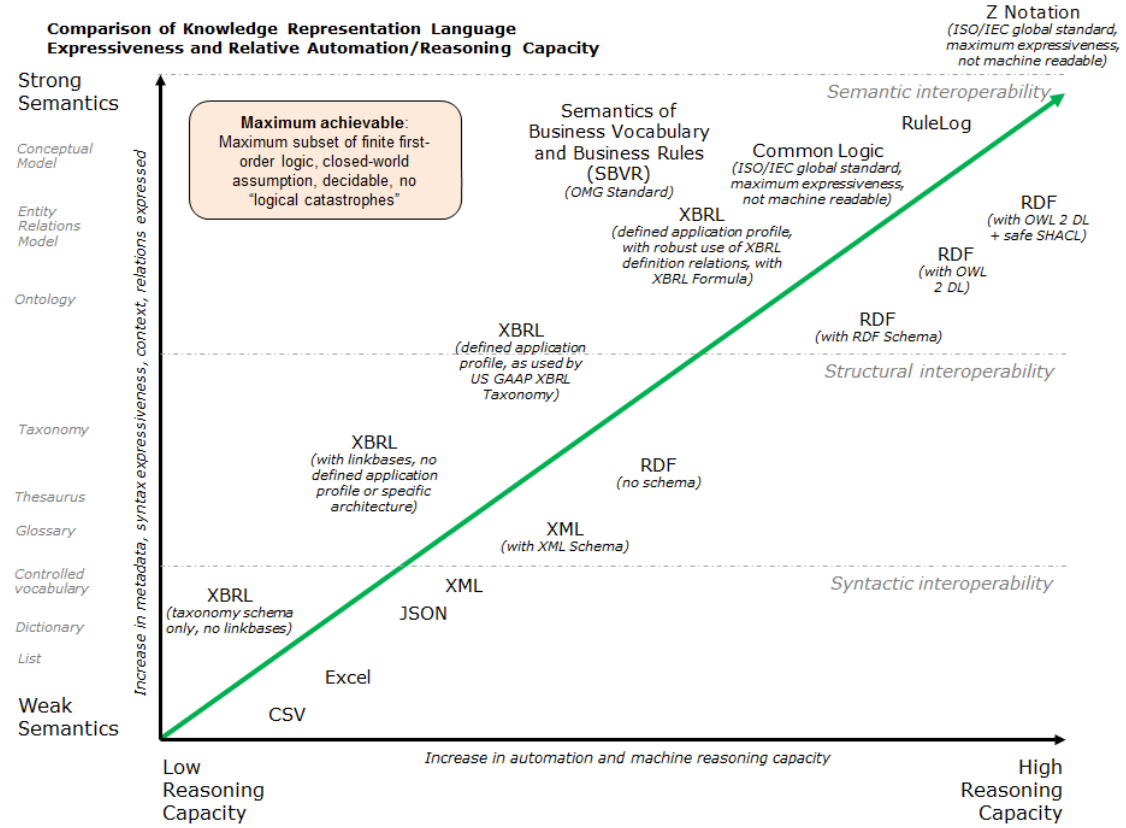
1.3.4. Understanding the relation between expressiveness and reasoning capacity

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 of the language, 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.



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, RuleLog, SHACL, and XBRL as having strong semantics. Those 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 expressiveness of the representation language used should be exposed so that the limitations become conscious.

1.3.5. Specific expressiveness features comparison

The following is a comparison of specific features of expressiveness[50]:

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	✓	✓	✓	×	×	×

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

1.3.6. 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:

- **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 for most business purposes must be decidable. Saying this another way, three-value logic[51] (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.

1.3.7. 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).

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

- 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. Relational databases use this 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.

1.3.8.Inference techniques

There are various types of inference techniques available[52].

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

First-order logic has limitations[53]. 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[54], 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?

---

[52] Decision Support Systems and Intelligent Systems, Efraim Turban and Jay E. Aronson 6th ed, Copyright 2001, Prentice Hall, Upper Saddle River, NJ, *Inference Techniques*, Chapter 13, http://www.indiana.edu/~bnwrbk/K510/ch13.ppt

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

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

1.3.10. 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 be a specific problem solving logic limited to business reporting and financial reporting.

1.3.11. Approaches for representing information logically in machine readable form

There are four 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.
- **Markup language**: A markup language of some sort.
- **Graphical format**: A graphical format.

1.3.12. Functional layers or categories of problem solving logic

Problem solving logics can be grouped into "functional layers" or "functional categories" or "functional groups"[55]:

- **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.

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

- **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.

1.3.13. 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.

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					

1.3.14. Summary of industry initiatives and standards

The following is a summary of industry initiatives and standards that can be used to implement business rules:

- **RuleML**: Rule Markup Language (RuleML) is an international industry initiative that closely collaborates with the W3C on the standardization of high-precision rules. RuleML is a family of closely related designs for different logical languages that overlap. (http://wiki.ruleml.org/index.php/RuleML_Home; http://cs.unb.ca/~boley/papers/RuleML-Overarching.pdf)

- **RIF**: Rules Interchange Format (RIF) is the W3C's design based on RuleML. RIF is a standard for exchanging rules among rule systems. W3C is an umbrella mature standards design organization. (http://www.w3.org/TR/rif-overview/)
- **RuleLog**: Rules Logical Programming Language (Rulelog) is an industry initiative to create a logical language designed to be appropriately expressive for supporting knowledge representation in complex domains, such as sciences and law, and yet to be efficiently implementable. (http://ruleml.org/rif/rulelog/spec/Rulelog.html)
- **Common Logic**: Common Logic is an ISO/IEC standard based on classical logic. (https://en.wikipedia.org/wiki/Common_Logic)
- **SBVR**: Semantics of Business Vocabulary and Business Rules (SBVR) is an OMG standard that was designed and built to be logically equivalent to Common Logic. (http://www.omg.org/spec/SBVR/)
- **SWRL**: Semantic Web Rule Language (SWRL) is an industry initiative, a special case of RuleML standards design. (http://www.w3.org/Submission/SWRL/)
- **SPIN**: SPARQL Inferencing Notation (SPIN) is an approach to writing rules using SPARQL notation; SPIN is similar to SWRL, a subset of RuleML, and a subset of RuleLog. SPIN is an industry initiative. (http://www.w3.org/Submission/spin-overview/)
- **SHACL**: SHACL (Shapes Constraint Language) is a language for validating RDF graphs against a set of conditions. Said another way, it is used for "expressing constraints on RDF data". And another way, "perform closed-world constraint checks on RDF-based data". SHACL replaces SPIN. SHACL is a W3C recommendation. (http://www.w3.org/TR/shacl/)
- **XBRL**: Extensible Business Reporting Language (XBRL) is an industry initiative standard specialized language for representing business report and financial report information including related rules. (https://www.xbrl.org/)

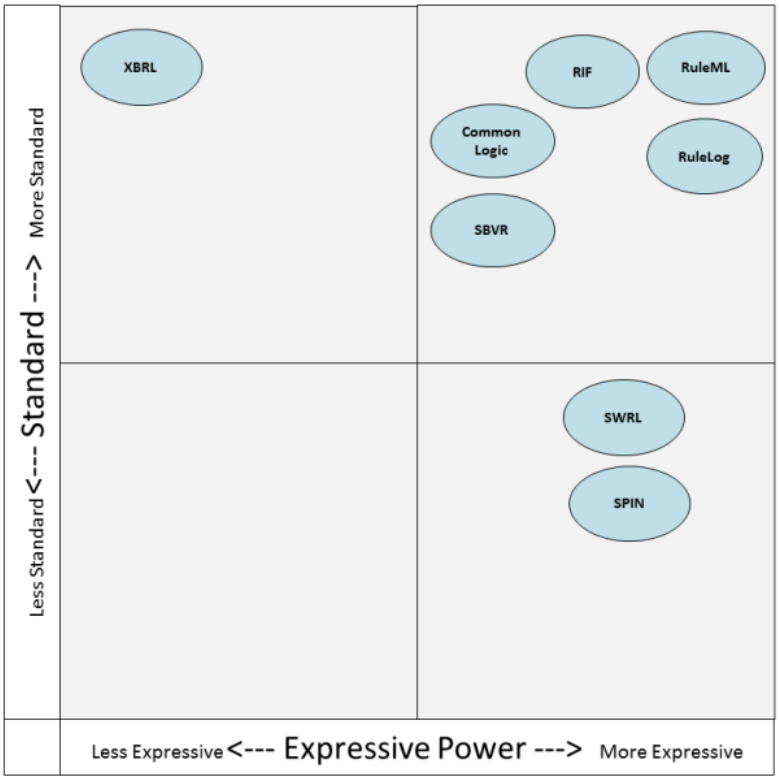
It appears that the "Semantic Web Stack[56]" will end up being RDF, OWL, and SHACL[57].

OWL was initially designed by well-intentioned academics. They missed that the open world assumption was a show stopper for business applications. SWRL and SIPN were industry initiatives to resolve issues with OWL. OWL 2 DL added the closed world assumption to OWL. SHACL was created to meet the needs shown by the industry initiatives SWRL and SPIN. SPIN is a subset of SHACL. SHACL achieved the right semantics for closed worlds.

The following graphic shows the relative expressive power and how standard each approach is based on what I have observed and learned:

[56] Wikipedia, *Semantic Web Stack*, https://en.wikipedia.org/wiki/Semantic_Web_Stack

[57] John Walker, https://twitter.com/wohnjalker/status/915982539747028992



The document *Survey of Knowledge Representations for Rules and Ontologies*[58] created by Benjamin Grosof provides additional details that are helpful in evaluating problem solving logic.

1.4. Implementing Problem Solving Logic in Software

You have probably heard that "computers are basically 1s and 0s". That is true. Implementing a problem solving logic is simply about managing the 1s and 0s.

1.4.1. NAND gate

The lowest denominator in implementing logic in software is what is called an NAND gate[59]. All logic systems can be converted into NAND gates. Theoretically, any logic function can be realized by correctly combining together enough NAND gates.

[58] Benjamin Grosof, *Survey of Knowledge Representations for Rules and Ontologies*, http://coherentknowledge.com/wp-content/uploads/2013/05/Ontolog-Forum-talk-OF-surveyKR-20131024-BNG.pdf

[59] Wikipedia, *NAND Gate*, retrieved June 7, 2017, https://en.wikipedia.org/wiki/NAND_logic

Q = NOT(**A** AND **B**)

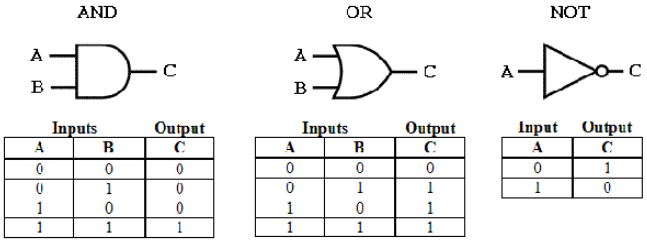
Truth Table

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

1.4.2.Low-level logical functions

Because there are many very common and general combinations of NAND gates that are used over, and over, and over; low-level logical functions are created to make representing that logic easier. Here are common low-level logical functions:

- AND
- NOT
- OR

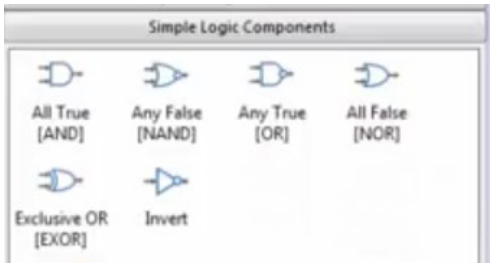


Inputs		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Input	Output
A	C
0	1
1	0

A function is logically just a combination of NAND gates. Lower level logical components are pieced together to provide the precise logic that you need. Commonly used logical functions are created to make this process easier. Getting software to perform the task that you want is simply piecing together the correct logical building blocks to arrive at the logic that you desire, the software doing precisely what you want the software to do.



1.4.3.Higher-level logical functions

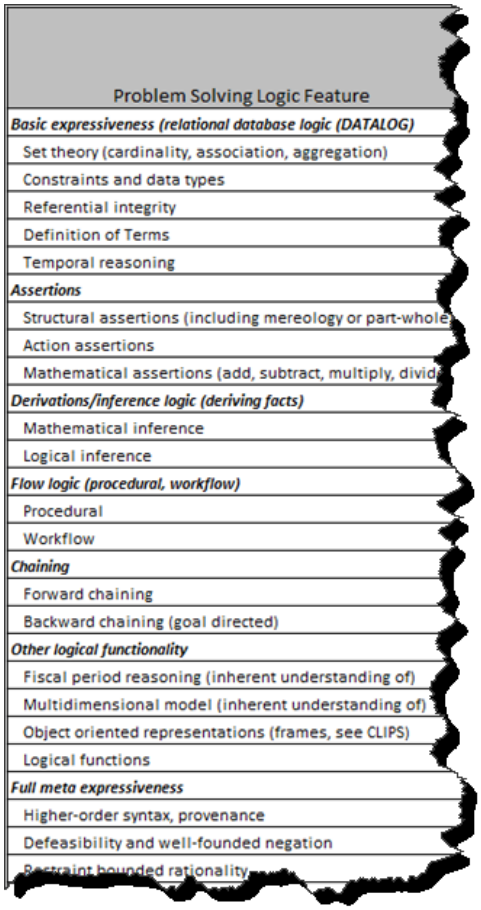
To make things even easier to implement, other layers of higher-level logical functions can be implemented. The more specific the set of functions, the easier the functions are to use but the less general the applicability of functions are to general

use cases. However, the more general the function or the lower the level of a function, the harder that function is to use.

Higher-level functions is logically just combinations of lower-level functions which have been combined together.

1.4.4.High-level problem solving logic features

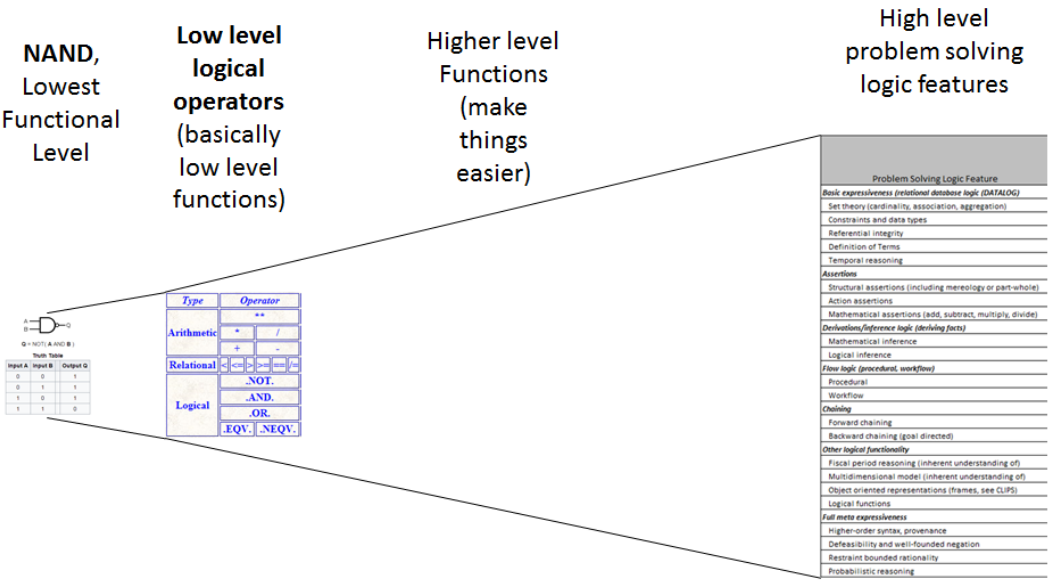
In a previous section, we pointed out high-level problem solving logic features that have general applicability to most problem domains. Problem solving logic features can be grouped into categories:



Problem solving logic features are simply sets of functions that provide specific problem solving logic functionality. Note that problem solving logic must be safely implementable so that the functionality provided is safe, predictable, repeatable, etc.

1.4.5.Summary of problem solving logic capability levels

The diagram below summarizes how lower-level problem solving logic is used to build higher-level problem solving logic features:



1.4.6.Implementation alternatives

Business professionals need problem solving logic to solve problems. There are three general areas where the problem solving logic can be implemented:

- Not implemented (i.e. the software user must manually perform the task)
- Implemented in software application (i.e. programmer implementing software creates)
- Implemented in a platform used by the software application (i.e. rules engine)

1.4.7.Make or buy decision

Each implementation alternative has pros and cons associated with it. Looking at the basket of pros and cons for each implementation alternative helps one choose the correct implementation alternative.

1.4.8.Implementation technology

An automobile is simply metal, plastic, rubber, and glass. The base components of an automobile are always the same; you have an engine, transmission, wheels, seats, body, etc.

However while a Hummer H2 and a Ford Fiesta are both automobiles, a Hummer H2 is a better off-road vehicle. Conversely, a Ford Fiesta is much better on the highway. Arguably, a motorcycle fits into the same set of base components as an automobile; and depending upon how you define "automobile" a motorcycle may or may not be properly categorized with a Hummer H2 and a Fort Fiesta.

You can drive either a Hummer H2, a Ford Fiesta, or a motorcycle off-road or on the highway. The question is, would you really want to do that.

So the question is: what is the best technology to use to implement a problem solving logic. Well, most software vendors would tell you that there product is the best product. Interesting how that tends to work!

You can construct pretty much any set of problem solving logic using any set of tools: RDF/OWL/SHACL and the semantic web stack[60], business rules management system[61], Microsoft Visual Basic, relational database management system, Java, MySQL, COBOL, XBRL Formula processor. The real question is: do you REALLY want to do that?

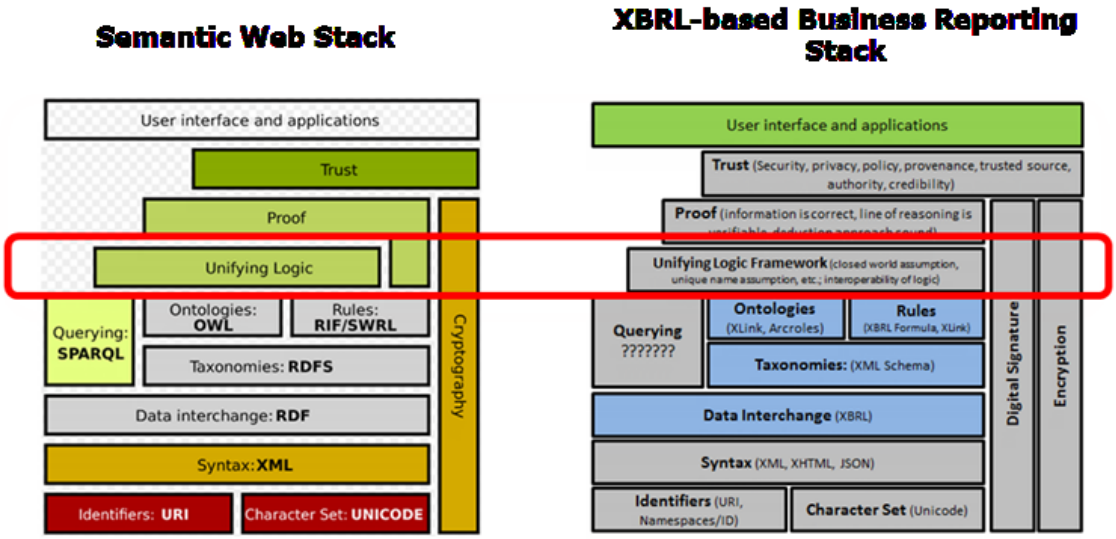
It all boils down to the set of pros and cons associated with each implementation approach and comparing those pros and cons with your needs. The better the match, the more effective and efficient your implementation will be.

1.5. Unifying Problem Solving Logic Framework for Business

What if there was one logic framework that everyone used for business applications? So, the logic framework is not the actual business logic. The framework is the logic related tools that one has to use to represent business logic. What if there was a ***Unified Problem Solving Logic Framework for Business***[62]?

1.5.1. Comparing and contrasting the Semantic Web Stack and XBRL Stack

Below is a comparison of the *Semantic Web Stack* on and a similar diagram of the *XBRL-based Business Reporting Stack*[63].



Ultimately systems need to interoperate. This is particularly true today in our networked world in our Digital Age. There is an intersection between these two stacks. That intersection is the "Unifying Logic" layer in the Semantic Web Stack and what I named the "Unifying Logic Framework" in the XBRL-based Business Reporting Stack in the diagram.

---

[60] Wikipedia, *Semantic Web Stack*, retrieved June 7, 2017, https://en.wikipedia.org/wiki/Semantic_Web_Stack

[61] Wikipedia, *Business Rules Management System*, retrieved June 7, 2017, https://en.wikipedia.org/wiki/Business_rule_management_system

[62] *Unified Logic Framework for Business*, http://xbrl.squarespace.com/journal/2017/11/26/unifying-logic-framework-for-business.html

[63] *Comparing and Contrasting Semantic Web Stack and XBRL Stack*, http://xbrlsite.azurewebsites.net/2017/Library/SemanticWebStack_XBRLStack.pdf

The common denominator in different technical implementation by business systems is business logic[64].

So, for example, the business logic that is referred to as the Accounting Equation[65] is that "Assets = Liabilities + Equity". That logic is the same in the Semantic Web Stack, or XBRL-based Business Reporting Stack, or any other technical implementation of any technology stack. It will never be the case that if business logic is to work one way in one system that the same business logic would work differently in some other system. Note that this is not the same thing as different systems having different business logic which is, by definition, different.

Another business logic rule with which accountants may not be as directly familiar, but they are indirectly familiar with this rule, notion of *closed world assumption*[66] or as contrast to the *open world assumption*[67]. The reason I say that professional accountants and other business professionals are indirectly familiar with this logic rule is because SQL databases all use the *closed world assumption* in the logic they use for answering questions of their users. The reason this is important to understand is that if one technical architecture was using the *closed world assumption* and another was using the *open world assumption* to answer questions there is a possibility of getting different answers to the exact same question from two different systems, which is an undesirable result.

Finally, any interaction between two systems is limited to the lowest common denominator in terms of the logic framework of the two systems. The reason is that it is only to the degree of logic provided by some logic framework that conveyers of information can represent the business rules that enforce business logic and therefore it is to that degree that quality can be effectively managed. Recall this from our previous discussion about relative expressiveness.

1.5.2.Advantages of a unifying problem solving logic framework

The following is a summary of the advantages of having one global standard problem solving logic framework:

- If business professionals interact with systems at the level of business logic, then the business professionals will **find the system approachable** and they will be able to effectively use the system because they are interacting using something they understand, business logic.
- Logic frameworks have different abilities to express business logic. The **MOST POWERFUL** logic framework that is **SAFE TO USE** (i.e. reliable, predictable, repeatable) is what business professionals tend to desire.
- If no one consciously creates some global standard **Unifying Logic Framework for Business**, then the logic frameworks could be incompatible between two business systems that are interacting and business professionals creating, consuming, or otherwise using information using those different systems might not be on par.

---

[64] *Common Denominator is Business Logic*, http://xbrl.squarespace.com/journal/2017/11/24/common-denominator-is-business-logic.html

[65] Wikipedia, *Accounting Equation*, https://en.wikipedia.org/wiki/Accounting_equation

[66] Wikipedia, *Closed World Assumption*, https://en.wikipedia.org/wiki/Closed-world_assumption

[67] Wikipedia, *Open World Assumption*, https://en.wikipedia.org/wiki/Open-world_assumption

- **Understanding the boundaries of a logic framework is important**; business professionals need to understand what the logic framework does and does not provide to most effectively employ the logic system.

In my personal view, XBRL International should consider creating such a **Unified Logic Framework for Business** in general, or I guess it could be a **Unified Logic Framework for Business Reporting** specifically. Because the Semantic Web Stack created by the W3C has a significantly more powerful logic framework that has, over the past 25 years been consciously tuned to be completely safe but very, very powerful for business systems and such applications; XBRL is at a significant disadvantage and, in my view, is at risk of becoming less relevant. The most powerful business systems will be those with the most powerful problem solving logic. One standard problem solving logic is better than many different proprietary problem solving logics.

1.5.3. Defining a unifying problem solving logic for business

To define some unifying problem solving logic for business, two alternative approaches exist: (a) use something that exists or (b) create something new. The current alphabet soup of standard logic frameworks for rules, candidates for such a unifying logic framework are perhaps:

- ISO/IEC Common Logic (CL)
- OMG Semantics of Business Vocabulary and Business Rules (SBVR)
- W3C RDFS + OWL + RIF/SWRL syntax logic (SWRL is not a recommendation, only a submission, RIF and SWRL seem to have issues)
- W3C RDFS + OWL + SHACL syntax logic which specifies closed world assumption and unique names assumption
- Industry Initiative RuleLog which is designed to be appropriately expressive for supporting knowledge representation in complex domains and yet to be efficiently implementable
- Industry Initiative RuleML which allows for partially constrained logic profiles and fully-specified logic semantics
- XBRL Formula (which has known deficiencies[68])

1.5.4. Characteristics of such a unifying problem solving logic framework

The following is a summary of the characteristics of such a unifying problem solving logic framework:

- a global standard logic framework for business
- represented using a controlled natural language format
- represent logic at the highest level possible such that the logic is understandable by business professionals
- rules created are approachable by business professionals

[68] *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

- built in but optional multidimensional model that does not force the use of OLAP, but usable with OLAP or OLTP
- enables interoperability between technology stacks
- enables interoperability between XBRL, Global Legal Entity Identifier (GLEI)[69], Financial Industry Business Ontology (FIBO)[70], Financial Regulation Ontology (FRO)[71], the US GAAP XBRL Taxonomy, the IFRS XBRL Taxonomy, etc.
- built based on the logic framework of the Semantic Web Stack which has been evolving for 25 or so years

[69] *Global Legal Entity Identifier*, https://www.gleif.org/en/about/this-is-gleif
[70] *Financial Industry Business Ontology*, http://www.edmcouncil.org/financialbusiness
[71] *Financial Regulation Ontology*, http://finregont.com/