1. Processing Logical Systems

This section provides information related to pragmatic processing machine-readable logical systems. Terms used to describe this include computational logic[1] and logic programming[2].

First, we explain the terms we will be using to describe a logical system. Next, we explain the notion of a properly functioning logical system. After that we provide a very simple example of a logical system to help you wrap your head around these ideas.

Finally, we talk about issues related to the physical processing of the machine-readable information described by the logical system.

1.1. Impact of Automation on Work

PWC predicts that "Global GDP will be 14% higher in 2030 as a result of AI – the equivalent of an additional $15.7 trillion. This makes it the biggest commercial opportunity in today's fast changing economy.[3]"

How work gets done will change. One Forbes article[4] explains that organizations are already leveraging AI in three common ways:

- Improving internal business processes
- Creating more intelligent products
- Offering a more intelligent service

Another Forbes article[5] points out how companies are delighting customers with artificial intelligence driven services:

- Delivering highly personalized offerings
- Giving customers more value
- Predicting customers needs

Without a doubt more and better human-machine collaboration is in store which will impact how work gets done within enterprises large and small.

Another example of the possibilities for financial accounting, reporting, and auditing is provided by the paper *Computational Law: The Cop in the Backseat*[6]. Intuit's

---

[1] Wikipedia, *Computational Logic*, https://en.wikipedia.org/wiki/Computational_logic

[2] Wikipedia, *Logic Programming*, https://en.wikipedia.org/wiki/Logic_programming

[3] PWC, *AI to drive GDP gains of $15.7 trillion with productivity, personalisation improvements*, https://www.pwc.com/gx/en/news-room/press-releases/2017/ai-to-drive-gdp-gains-of-15_7-trillion-with-productivity-personalisation-improvements.html

[4] Bernard Marr, Forbes, *3 Important Ways Artificial Intelligence Will Transform Your Business And Turbocharge Success*, https://www.forbes.com/sites/bernardmarr/2020/08/03/3-important-ways-artificial-intelligence-will-transform-your-business-and-turbocharge-success/

[5] Bernard Marr, Forbes, *3 Huge Ways Companies Are Delighting Customers With Artificial-Intelligence-Driven Services*, https://www.forbes.com/sites/bernardmarr/2020/08/21/3-huge-ways-companies-are-delighting-customers-with-artificial-intelligence-driven-services

[6] Michael Genesereth, Stanford University Center for Legal Informatics, *Computational Law: The Cop in the Backseat*, http://logic.stanford.edu/complaw/complaw.html

Turbotax is an example of a computational law system. Millions use Turbotax each year to prepare their tax forms. Statutory and regulatory financial accounting, reporting, and auditing rules are essentially laws.

1.2. Financial Report is Man-made Logical System

But how exactly does this automation actually work? Financial reporting offers insight into this question.

With any field of knowledge, the critical concepts of the field (a.k.a. jargon) are embedded in the definitions of the field's technical terms. The term 'statement' in financial reporting is different than that same term 'statement' as is being used here. Here, the term 'statement' is used in the propositional logic sense[7].

A financial report is a man-made logical system. A financial report is a definite set of statements that explain the financial position and financial performance of an economic entity. In the past, these statements were communicated using the medium of paper which is only readable by humans. But now, the global standard XBRL and other such technologies enables financial reports to be not only machine-readable but also machine-understandable, machine-executable, and machine-interpretable. As a result, effective automation can be achieved.

While technology is certainly an enabler; a Harvard Business Review article[8] points out that harnessing the power of information is more about talent than it is about technology.

To understand how all this can work, lets first take a step back and look at a few important details about the financial report logical system. This example will provide insight into how other business reports and other business systems can and likely will be automated.

The first step is to understand the notion of a logical system.

1.3. Logical System Explained in Non-technical Terms

A **system**[9] is a cohesive conglomeration of interrelated and interdependent parts that is either natural or man-made[10]. When a system is working right, it creates a virtuous cycle[11].

A **pattern** is any form of correlation between the states of elements within a system. Patterns are structural prescriptions to which model artifacts are meant to conform.

A **theory**[12] is a tool that can be used to describe a system. A theory is a set of statements in a formal language. A theory essentially describes the patterns within

---

[7] Internet Encyclopedia of Philosophy, *Propositional Logic*, https://iep.utm.edu/prop-log/

[8] Becky Frankiewicz and Tomas Chamorro-Premuzic, Harvard Business Review, *Digital Transformation Is About Talent, Not Technology*, https://hbr.org/2020/05/digital-transformation-is-about-talent-not-technology

[9] Wikipedia, *Systems Theory*, https://en.wikipedia.org/wiki/Systems_theory

[10] Charles Hoffman, CPA, *Systems Theory: Method to my Madness*, http://xbrl.squarespace.com/journal/2019/12/29/systems-theory-method-to-my-madness.html

[11] *Virtuous Cycle*, http://xbrl.squarespace.com/journal/2020/4/29/virtuous-cycle.html

[12] Wikipedia, *Theory (Mathematical)*, https://en.wikipedia.org/wiki/Theory_(mathematical_logic)

a system. For example, the *Logical Theory Describing Financial Report*[13] describes the logical patterns found in financial reports.

A **proof**[14] is verification that shows that all the statements within a theory are clearly and unmistakably true.

As such, a logical system can be explained by a logical theory. A logical theory is an abstract conceptualization[15] of specific details of some domain. The logical theory provides a way of thinking about a domain by means of deductive reasoning to derive logical consequences of the theory. (a.k.a. axiomatic system[16])

A **logical theory** enables a community of stakeholders trying to achieve a specific goal or objective or a range of goals/objectives to agree on important statements used for capturing meaning or representing a shared understanding of and knowledge in some universe of discourse.

A logical theory is made up of a set of *models*, *structures*, *terms*, *associations*, *rules*, and *facts*[17]. In very simple terms,

- **Logical theory**: A *logical theory* is a set of models that are consistent with and permissible per that logical theory.
- **Model**: A *model*[18] is a set of structures that are consistent with and permissible interpretations of that model.
- **Structure**: A *structure* is a set of statements which describe the structure. A structure is a composite. A structure is used to partition fragments of a model.
- **Statement**: A statement is a proposition, claim, assertion, belief, idea, or fact about or related to the universe of discourse to which the logical theory relates. There are four broad categories of statements:
    - **Terms**: Terms are statements that define ideas used by the logical theory such as "assets", "liabilities", "equity", and "balance sheet". A term is a type of statement that specifies the existence of a primitive (a.k.a. simple, atomic) or functional (a.k.a. complex, composite) idea that is used within a universe of discourse. Terms are generally nouns. (Tbox[19])
    - **Associations**: Associations (a.k.a. relation, predicate) are statements that describe permissible interrelationships between the terms such as "assets is part-of the balance sheet" or "operating expenses is a type-of expense" or "assets = liabilities + equity" or "an asset is a 'debit' and is 'as of' a specific point in time and is always a monetary numeric

[13] *Logical Theory Describing Financial Report*, http://xbrl.squarespace.com/logical-theory-financial-rep/
[14] Richard Hammack, *Book of Proof*, Chapter 14 Direct Proof, page 113, https://www.people.vcu.edu/~rhammack/BookOfProof/Main.pdf
[15] Wikipedia, *Conceptual Model*, https://en.wikipedia.org/wiki/Conceptual_model
[16] Wikipedia, *Axiomatic System*, https://en.wikipedia.org/wiki/Axiomatic_system
[17] *Understanding and Expressing Logical Systems*, http://xbrl.squarespace.com/journal/2019/9/25/understanding-and-expressing-logical-systems.html
[18] Wikipedia, *Model Theory*, https://en.wikipedia.org/wiki/Model_theory
[19] Wikipedia, Tbox, https://en.wikipedia.org/wiki/Tbox

value". An association is a type of statement that specifies a permissible structure or specifies a property of a term. Associations are generally a verbs.

- **Is-a**: An is-a association specifies a general-special[20] or wider-narrower or type-subtype[21] type relation between terms.
- **Has-a**: A has-a association specifies a has-part or part-of type relation between terms. (meronymy[22])(composition[23])
- **Property-of**: A property-of association specifies that a term has a specific quality, trait, or attribute. (property[24])

- **Rules**: Rules (a.k.a. assertions) are statements that specify a permissible modification within a model or structure. For example, "IF the economic entity is a not-for-profit THEN net assets = assets - liabilities; ELSE assets = liabilities + equity". (Abox[25])
  - **Axiom**: An axiom[26] (a.k.a. postulate) is a statement which describes a self-evident logical principle related to a universe of discourse that no one would argue with or otherwise dispute.
  - **Theorem**: A theorem[27] is a statement which makes a logical deduction which can be proven by constructing a chain of reasoning by applying axioms or other theorems in the form of IF…THEN statements.
  - **Restriction**: A restriction (a.k.a. constraint) is a statement that is a special type of axiom or theorem imposed by some authority which restricts, constrains, limits, governs, or imposes some range or otherwise specifies integrity (i.e. correctness).
- **Facts**: Facts are statements about the numbers and words that are provided by an economic entity within a business report. For example, the financial report, a type of business report, might state "assets for the consolidated legal entity Microsoft as of June 20, 2017 was $241,086,000,000 expressed in US dollars and rounded to the nearest millions of dollars.

Connectors are used to join one or more well-formed statements and include:

- Implication
- Disjunction (or); exclusive disjunction (xor)
- Conjunction (and)
- Negation (not)

---

[20] Caminao, *Specialization vs Generalization*, https://caminao.blog/what-who-how-when-where/how-to-represent-objects-and-activities/quality/specialization/
[21] Giuseppe Castagna and Alain Frisch, *A Gentle Introduction to Semantic Subtyping*, https://www.irif.fr/~gc/papers/icalp-ppdp05.pdf
[22] Wikipedia, *Meronymy*, https://en.wikipedia.org/wiki/Meronymy
[23] Wikipedia, *Composition*, https://en.wikipedia.org/wiki/Class_diagram#Composition
[24] Wikipedia, *Property (Mathematics)*, https://en.wikipedia.org/wiki/Property_(mathematics)
[25] Wikipedia, *Abox*, https://en.wikipedia.org/wiki/Abox
[26] Wikipedia, *Axiom*, https://en.wikipedia.org/wiki/Axiom
[27] Wikipedia, *Theorem*, https://en.wikipedia.org/wiki/Theorem

- Logical equivalence (if and only if)

Qualifiers is used to extend propositional logic[28] into predicate logic[29] and include:

- There exists (existential qualifier)
- For all (universal qualifier)

And so, fundamentally, a logical theory is a set of statements. Those statements must follow a logic. Those statements can be represented in machine-readable form. Those statements can be proven to be consistent with one another. Once in machine-readable form, those statements can be interrogated using software applications. To the extent that this can be done effectively; software tools can assist professional accountants and others working with those statements.

A financial report has a finite set of statements (structures, terms, associations, rules, and facts) within the report. The set of statements is definite. That definite set of statements forms a model.

There seems to be many fundamentally different ways (terminology) to describe the essence of what I am describing in this section. Perhaps I got a few details incorrect. This is not a problem; I can simply correct each of those mistakes until every detail is, in fact, correct. It is pretty much impossible to dispute the essence of what I am trying to get at because logic, mathematics, and computer science are built upon this same foundation.

1.4. Proper Functioning Logical System

A logical theory is said (or proven) to be **consistent** if there are no contradictions with respect to the statements made by the logical theory that describes the logical system (i.e. reality).

Three categories of errors can occur:

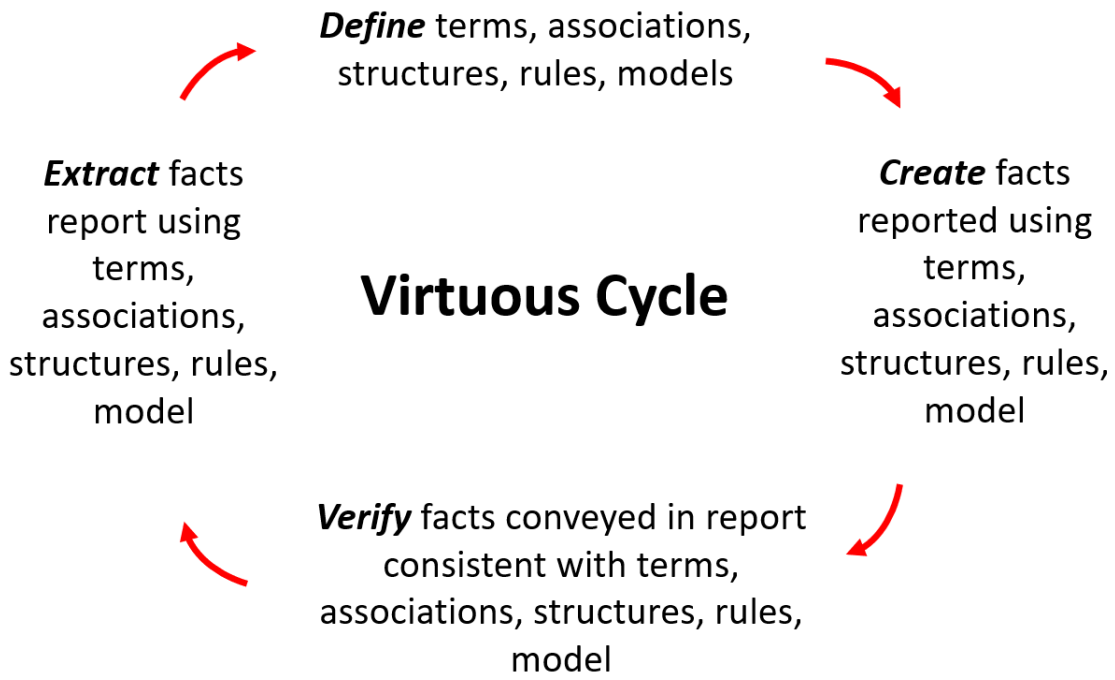
- **Syntax errors**: A syntax error in a logical system is similar to computer code not being able to compile. For example, when an XBRL processor tells you that your XBRL taxonomy is not valid per the XBRL technical specification, that is a syntax error.
- **Logic errors**: A logic error within a logical system is where the machine-readable logic is configured in a manner that is inconsistent with what exists in reality and causes the logical system to not work as expected. For example, if you represented something in your XBRL taxonomy as a credit when it should have been a debit that is a logic error. There are two categories of logic errors:
  - **Model logic error**: A model is not consistent with the prescribed meta-model specification.
  - **Instance logic error**: A statement made within an instance contradicts or is otherwise inconsistent with some other statement.
- **Precision and coverage errors**: *Precision* is a measure of how precisely the information within a logical theory has been represented as contrast to reality

[28] Wikipedia, *Propositional Logic*, https://en.wikipedia.org/wiki/Propositional_calculus
[29] Wikipedia, *Predicate Logic*, https://en.wikipedia.org/wiki/First-order_logic

of the logical system for the universe of discourse. *Coverage* is a measure of how completely information in a logical theory has been represented relative to the reality of the logical system for a universe of discourse.

A logical theory can have high to low **precision** and high to low **coverage**.

When a logical system is consistent and it has high precision and high coverage the logical system can be considered a properly functioning logical system. When a system is working right, it creates a virtuous cycle[30].



1.5. Very Simple Example of Logical System

A very simple example of a logical system is the accounting equation. Here is a description of the accounting equation logical system in both human-readable terms and machine-readable terms using XBRL[31]:

Terms: Three simple terms are defined: Assets, Liabilities, Equity. One complex term is defined, balance sheet.

Structure: One structure is defined, the balance sheet, and identified using the term balance sheet.

Associations: The three terms Assets, Liabilities, and Equity are associated in that they are all PART-OF the structure balance sheet.

Rules: A mathematical assertion is made that "Assets = Liabilities + Equity".

Facts: Instances of three facts are established to exercise the model: Assets of $5,000; Liabilities of $1,000; Equity of $4,000.

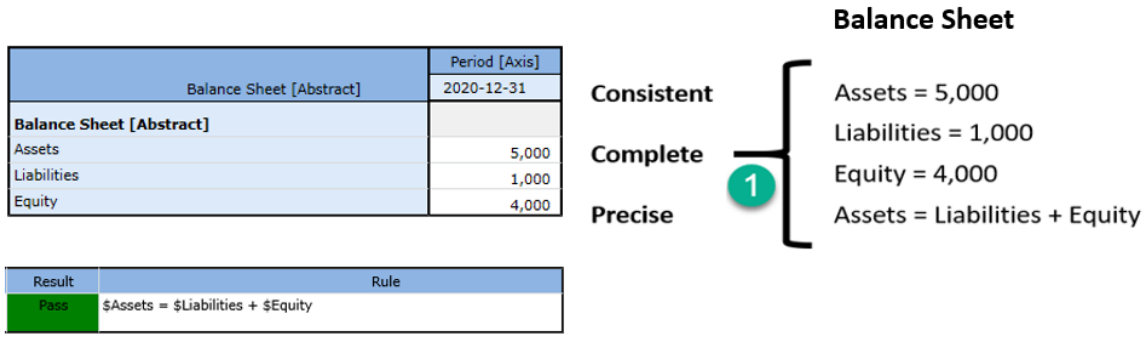
---

[30] Charles Hoffman, CPA, *Virtuous Cycle*, http://xbrl.squarespace.com/journal/2020/4/29/virtuous-cycle.html

[31] Charles Hoffman, *Accounting Equation*, http://xbrlsite.azurewebsites.net/2020/master/ae/

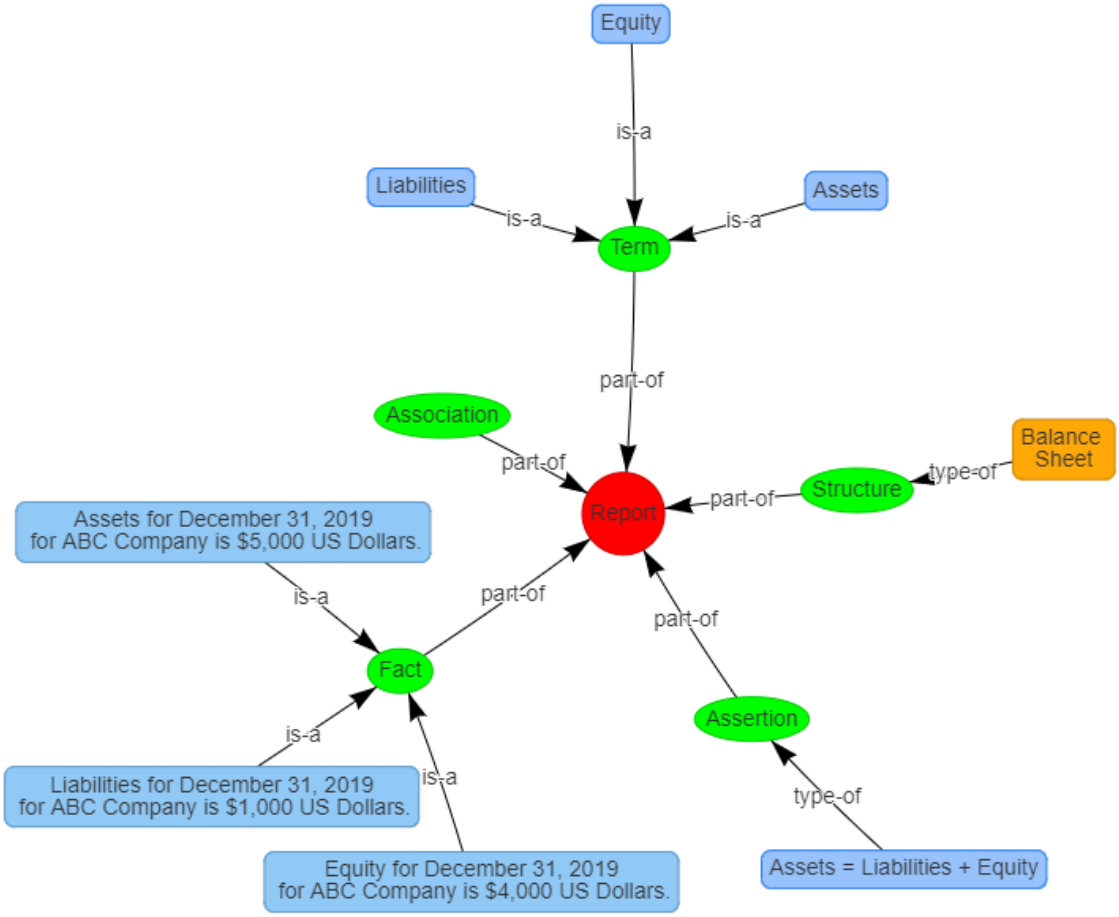
Model: All of the terms, associations, assertions, structures, and facts describe the model. We created only one model, or permissible interpretation, of the logical theory. (As accountants know, if you reverse the equation using the rules of math to "Equity = Assets - Liabilities" and change the term "Equity" to "Net Assets"; then you get another permissible interpretation or model.)

Because this is a very simple example with only a few statements it is easy to get your head around this system and see that it is consistent, complete, and precise. As expected, you see three facts described by three terms which are related to one structure and the one rule is consistent with expectation:



As the size of the logical system increases it becomes increasingly more challenging to verify that the logical system is properly function using manual processes. But, covering the impediments to a properly functioning logical system are beyond our scope here[32]. Essentially, the models, terms, structures, rules, and facts form a directed acyclic graph such as:

[32] Charles Hoffman, CPA, *Impediments to Creating Properly Functioning XBRL-based Reports*, http://xbrlsite.azurewebsites.net/2020/core/master-ae/Documentation.pdf



While a typical financial report is significantly larger (i.e. the Microsoft 2017 10-K is made up of 194 structures) every financial report works the same as this very simple example but just has more pieces.

1.6. Summary: Building on the Shoulders of Giants

A logical system[33] is a type of formal system[34]. A financial report is a type of formal system. To be crystal clear what I am trying to create is a **finite model-based deductive first-order logic system**[35]. "Finite" as opposed to "infinite" because finite systems can be explained by math and logic, infinite systems cannot. "Model-based" is the means to address the necessary variability inherent in the required system. "Deductive", or rule-based, as contrast to inductive which is probability based which is not appropriate for this task. "First-order logic" because first-order logic can be safely implemented within software applications and higher order logics are unsafe. "System" because this is a system.

The point is to create a logical system that has high expressive capabilities but is also a provably safe and reliable system that is free from catastrophic failures and

[33] Wikipedia, *Logical Systems*, https://en.wikipedia.org/wiki/Logic#Logical_systems
[34] Wikipedia, *Formal System*, https://en.wikipedia.org/wiki/Formal_system
[35] Wikipedia, *First-order Logic, Deductive System*, https://en.wikipedia.org/wiki/First-order_logic#Deductive_systems

logical paradoxes which cause the system to completely fail to function. To avoid failure, computer science and knowledge engineering best practices seems to have concluded that the following alternatives are preferable:

- **Systems theory**: A system[36] is a cohesive conglomeration of interrelated and interdependent parts that is either natural or man-made. Systems theory explains logical systems.
- **Logical theory**: (a.k.a. logical system) There are many approaches to representing "ontology-like things" in machine-readable form[37], a logical theory being the most powerful.
- **Proof theory**: The ideas of proof theory[38] can be used to verify the correctness of logical systems and computer programs working with those machine-readable logical systems.
- **Set theory**: Set theory is foundational to logic and mathematics. Axiomatic (Zermelo–Fraenkel) set theory[39] is preferred to naïve set theory.
- **Graph theory**: Directed acyclic graphs[40] are preferred to less powerful "trees" and graphs which contain cycles that can lead to catastrophic problems caused by those cycles.
- **Logic**: Logic is a formal communications tool. **Horn logic**[41] is a subset of first-order logic which is immune from logical paradoxes should be used as contrast to more powerful but also more problematic first order logic features. Note that deductive reasoning is leveraged for the process of creating a financial report and not inductive reasoning (i.e. machine learning)
- **Model theory**: Model theory is a way to think about flexibility. Safer finite model theory[42] is preferable to general model theory.
- **World view**: The following are common issues which appear when implementing logical systems in machine-readable form, the safest and most reliable alternatives are:
    - closed world assumption[43] which is used by relational databases is preferred to the open world assumption which can have decidability issues;
    - negation as failure[44] should be explicitly stated;
    - unique name assumption[45] should be explicitly stated;

Business professionals are (a) not capable of having precise discussions of these sorts of issues with software engineers, (b) don't care to have such technical

[36] Wikipedia, *Systems Theory*, https://en.wikipedia.org/wiki/Systems_theory
[37] Difference between Taxonomy, Conceptual Model, Logical Theory, http://xbrl.squarespace.com/journal/2018/12/11/difference-between-taxonomy-conceptual-model-logical-theory.html
[38] Stanford University, *The Development of Proof Theory, The Aims of Proof Theory*, https://plato.stanford.edu/entries/proof-theory-development/#AimProThe
[39] Wikipedia, *Set Theory, Axiomatic Set Theory*, https://en.wikipedia.org/wiki/Set_theory#Axiomatic_set_theory
[40] Wikipedia, *Directed Acyclic Graph*, https://en.wikipedia.org/wiki/Directed_acyclic_graph
[41] Wikipedia, *Horn Logic*, https://en.wikipedia.org/wiki/Horn_clause
[42] Wikipedia, *Finite Model Theory*, https://en.wikipedia.org/wiki/Finite_model_theory
[43] Wikipedia, *Closed World Assumption*, https://en.wikipedia.org/wiki/Closed-world_assumption
[44] Wikipedia, *Negation as Failure*, https://en.wikipedia.org/wiki/Negation_as_failure
[45] Wikipedia, *Unique Name Assumption*, https://en.wikipedia.org/wiki/Unique_name_assumption

discussions about these sorts of issues with software engineers, (c) are not interested in the theoretical or philosophical or religious debates that commonly exist related to these alternatives, (d) if the alternatives were ***appropriately articulated to a business professional***, who tend to be very practical, they would **most often error on the side of safety and reliability**.

1.7. Method

I have been able created a method[46] that can be effectively used to implement XBRL-based digital financial reporting using the logical framework described. Further, I have been able to prove that the method works effectively and have been able to convert 100% between logically compatible implementations using XBRL, Prolog, and Cypher[47]. An RDF + OWL + SHACL implementation is forthcoming.

The method supports the notion of a sanctioned inference. A "sanctioned" or "authorized" inference is defined as an inference deemed as an appropriate conclusion to draw from the explicitly provided information available.

1.8. Evaluating Implementation Alternatives

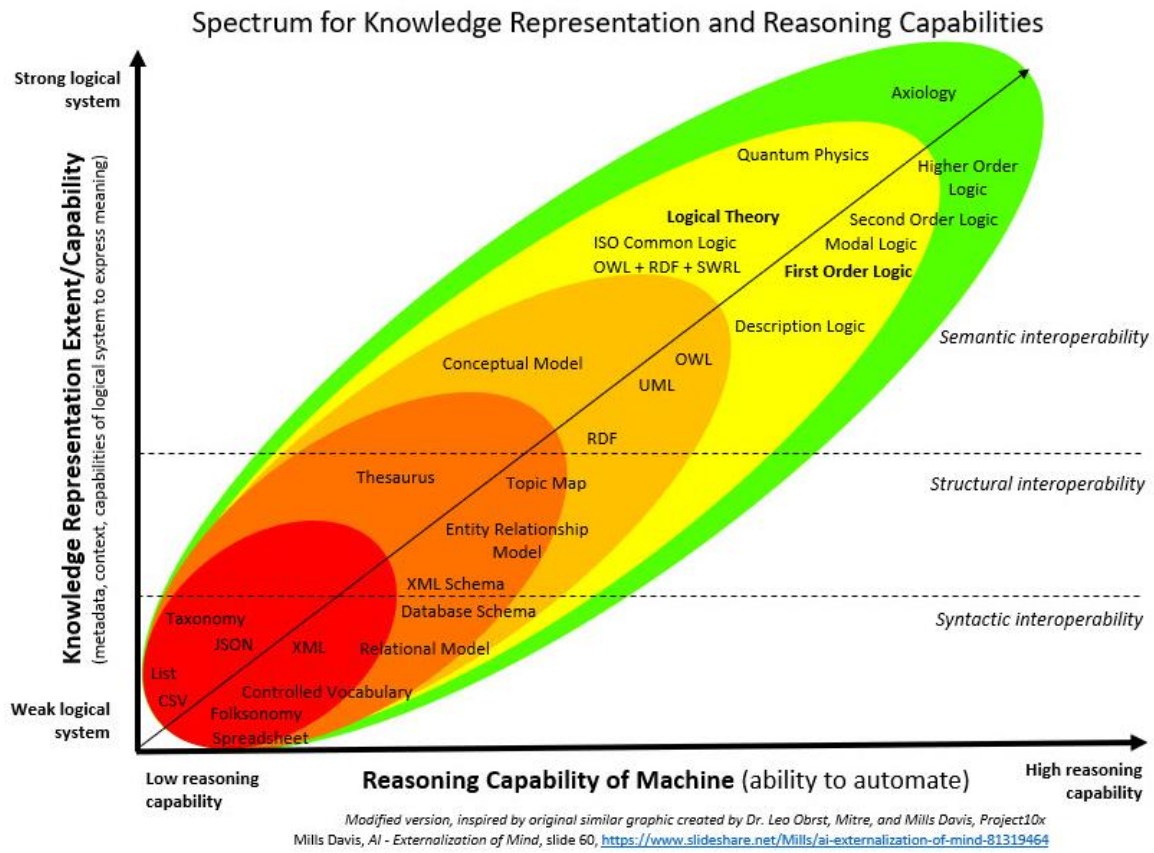
Figuring out which logic to use is a "dance" between expressively[48] and tractability[49], trying to get the right equilibrium for the task being performed. On the one hand, you want the logic you use to be as powerful as possible. But on the other hand, you need the logical system to be reliable, controllable, and not subject to catastrophic failures such as getting lost within an infinite loop.

So, there are inherent trade-off between expressiveness and tractability. Striking the right balance can be challenging and takes world-class level talent. One needs to be practical when trying to strike the appropriate balance and the balance should be driven by the task being performed. Nothing has both the maximum expressively and maximum tractability. Each alternative is a "basket" of functionality.

What I do know for certain is that an "ontology" alone is not enough because you cannot represent mathematical computations using an ontology and financial reports contain mathematical computations. You need "ontology + rules". SQL alone does not seem to be enough but I am not 100% sure. XBRL Formula is not enough, certain specific things are missing.

---

[46] Charles Hoffman, CPA, *Understanding Method*, http://xbrlsite.azurewebsites.net/2020/Library/UnderstandingMethod.pdf
[47] Charles Hoffman, CPA, *Understanding Proof*, http://xbrlsite.azurewebsites.net/2020/Library/UnderstandingProof.pdf
[48] *Reasoning Capacity*, http://xbrlsite.azurewebsites.net/2019/Library/ReasoningCapacity.jpg
[49] *Tractability*, https://www.dictionary.com/browse/tractability



For a through discussion of the issues of knowledge representation please see *What is a Knowledge Representation?*[50]

1.9. Criteria for Evaluating Between Implementation Alternatives

In the hands of a craftsman that knows what they are doing, is a specific implementation alternative tractable? Is it that you just need to be conscious about what you are doing, be aware of potential problem areas, and engineer around those potential problems? Is it the case that the advantages of an alternative is too compelling and the disadvantages so "small" (not sure of the word) that the alternative has its place? What are the appropriate evaluation criteria?

Understanding how to evaluate different implementation and rule/logic processing alternatives can be challenging. Things that seem important are:

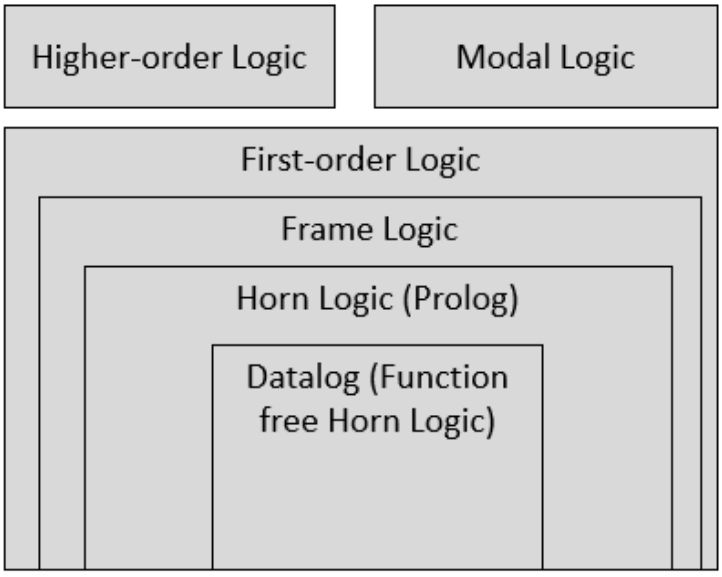
1. Is the processor "Turing-complete[51]"? (i.e. you can create a valid Turing Machine[52])
2. Is the logical system logically decidable[53]?

[50] Randall Davis, Howard Shrobe, Peter Szolovits, *What is a Knowledge Representation*, http://groups.csail.mit.edu/medg/ftp/psz/k-rep.html
[51] Wikipedia, *Turing Completeness*, https://en.wikipedia.org/wiki/Turing_completeness
[52] Wikipedia, *Turing Machine*, https://en.wikipedia.org/wiki/Turing_machine

3. What are the ramifications if a program is non-terminating[54]? (Church-Turing Thesis[55])
4. Is the processor susceptible to catastrophic failure?
5. Can logical paradoxes be avoided?
6. Can all needed semantics be expressed effectively?
7. What are the processing capabilities; do they meet all requirements?

Here is an example of the issues. One would think that understanding Prolog is rather straight forward. But, did you realize that there is a "Full Prolog[56]" and that there is a notion of "Pure Prolog"? In fact, there is a third Prolog[57], "Database Prolog".

As I understand it, Prolog is based on Horn Logic and therefore is not really a full PROgramming LOGic. Datalog is a subset of Prolog which excludes "functions" which can cause problems when trying to use Prolog with SQL databases. The notion of "Full Prolog" would be a complete set of first order logic.



Here is how the difference between Full Prolog, Pure Prolog, and Database Prolog was explained to me:

> With "pure" code, we mean that certain logical properties hold about the code. For example, adding a clause to the program can make the program at most more

[53] Wikipedia, Decidability (Logic), https://en.wikipedia.org/wiki/Decidability_(logic)
[54] Ramifications, http://www.philipp.ruemmer.org/publications/non-termination.pdf
[55] Stanford Encyclopedia of Philosophy, *The Church-Turing Thesis*, https://plato.stanford.edu/entries/church-turing/
[56] Egon Börger and Dean Rosenzweig, *A Mathematical definition of full prolog*, https://www.sciencedirect.com/science/article/pii/016764239500006E
[57] Subsets of Prolog, https://www-users.york.ac.uk/~sjh1/courses/L334css/complete/complete2se3.html

> general, never more specific. Likewise, adding a goal to a clause can make the program at most more specific, never more general.
>
> Thus, the pure subset of Prolog allows declarative reasoning, based on logical properties. It lets you generate explanations automatically; it lets you easily analyze program fragments, and it lets you easily parallelize programs, among other benefits.
>
> With "impure" Prolog code, we mean code that violates these properties. For example, Prolog code that changes files, or that emits output, is impure: We cannot freely reorder goals that print text on the terminal, because reordering goals changes the effect of the program. This is in contrast to pure code: In logic, conjunction is commutative, and in the pure subset of Prolog, we can freely reorder the goals of a conjunction without changing the meaning of the program:
>
> (A,B) means the same thing as (B,A).
>
> A very important research question about Prolog is: How can we increase the pure parts of Prolog, while retaining acceptable performance? Scryer Prolog already has several important and new constructs in this direction. For example, with Scryer Prolog, you can read from files in a pure way, by first stating by logical means what you expect from the file contents, and then applying a predicate that satisfies all logical properties we expect from relations to parse from an external file.
>
> So, these are not fixed sets: The pure part of Prolog is becoming ever more expressive and useful, as more declarative constructs are found and implemented. This is work in progress, it is the main drive behind a lot of research about Prolog implementation and standardisation. In the future, we expect the pure part of Prolog to become so expressive that very few impure constructs, if any, will be needed at all.
>
> Note that as soon as you allow Horn clauses, the language is already Turing-complete. So, both pure Prolog and "full" Prolog allow non-terminating programs, and queries that are no longer decidable. That's not catastrophic: In practice, often only very simple reasoning is needed, and for rule-based reasoning, a small fragment of Prolog is often already useful. For instance, to replace SQL databases, only Datalog is needed, and that is both decidable and very efficient.

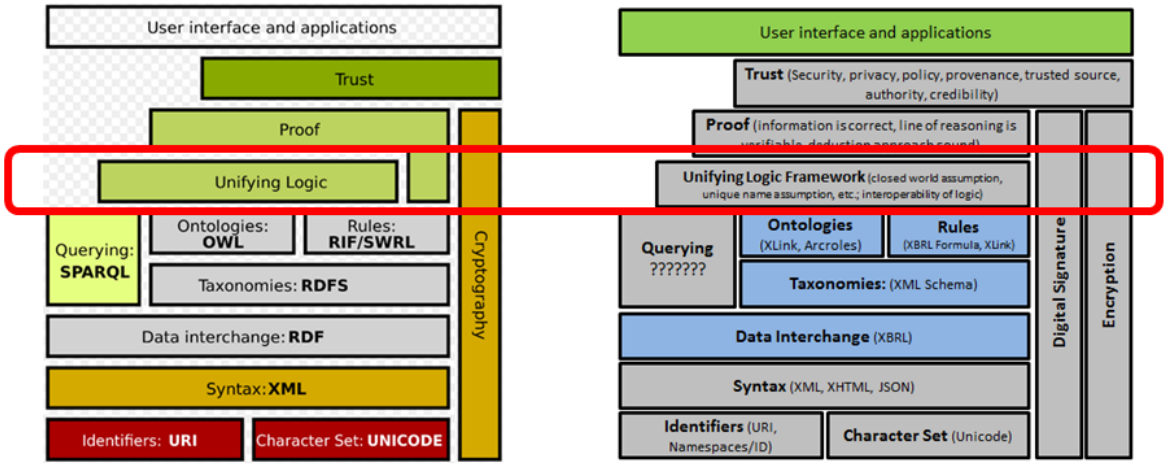
Personally, I really don't understand 100% of this, but testing and examples can make all of the above understandable.

1.10. Multiple Technology Stacks and Implementation Preferences

Arbitrary preferences, fads, trends, misinformation, and many other things influence how information technology professionals solve the same problem. As such, there will always be multiple technology stacks as opposed to every information technology professional using exactly the same approach to solving what amounts to be exactly the same problem.

Consider the comparison of these two technology architecture stacks[58]: Semantic Web Stack and XBRL Stack:

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



Each of the technical architecture stacks has a problem solving logic. The Semantic Web Stack on the left calls it a "Unifying Logic" and the XBRL Stack calls it a "Unifying Logic Framework"[59]. Other technology stacks also exist.

The problem solving logic for the Semantic Web Stack is defined by the technical syntax offered by RDFS, OWL, RIF/SWIRL, SHACL. For the Semantic Web Stack, those technical syntaxes provide the boundaries for the problem solving logic. For the XBRL Stack, there really are no real boundaries outlined at all for the problem solving logic. Even if someone could provide the specifics of the problem solving power offered by RDFS, OWL, and RIF/SWIRL, SHACL; that list undoubtedly would not be understandable to professional accountants or other business professionals. But if business professionals are supposed to control and maintain business rules, then those business logic boundaries would best be clear and understandable. The bottom line here is that the boundaries of a problem solving logic should be understandable.

1.11. Implementation Alternatives

There are two implementation alternatives that I am 100% certain will not work. First, OWL alone will not work because you cannot express mathematical computations in OWL. Second, XBRL Formula will not work as specified today because specific known deficiencies in functionality exist[60].

My confidence is pretty high that all of the following seem to provide enough logic, but each also has specific known control issues associated with them:

- **Ontology + Rules**: For example, OWL[61] + SHACL[62] + RDF[63] provides a sufficient fragment of first order logic. (Some call this Modern Symbolic AI[64])

[59] *Unifying Logic Framework for Business*, http://xbrl.squarespace.com/journal/2017/11/26/unifying-logic-framework-for-business.html
[60] 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
[61] W3C, *OWL*, https://www.w3.org/TR/owl2-overview/
[62] W3C, *SHACL*, https://www.w3.org/TR/shacl/
[63] W3C, RDF, https://www.w3.org/RDF/

- **Modern Prolog**: Prolog such as SWI Prolog[65] or Scryer Prolog[66] seems to have all of the necessary functionality. The up side is that there are a lot of Prolog implementations[67]. The down side is that none of these Prologs can call it self "the standard". Each has pros and cons. Some people seem to refer to this as "Pure Prolog" which is limited to Horn Logic[68]. Prolog works with relational (SQL) and graph databases.
- **ISO Prolog**: ISO has created a standard Prolog[69]. This appears to be a subset of Pure Prolog. There is solid motivation for implementations to support ISO Prolog, many already do to one degree or another.
- **Datalog**: Datalog[70], or "function-free Horn Logic", is more tractable than Pure Prolog and ISO Prolog. RuleML.org points out[71], "Datalog is the language in the intersection of SQL and Prolog. It can thus be considered as the subset of logic programming needed for representing the information of relational databases, including (recursive) views." So Datalog works with relational databases and graph databases[72] (more precisely labeled property graphs).
- **PSOA**: PSOA[73] (Positional-Slotted Object-Applicative) which some people refer to as "Full Prolog". PSOA works better than Datalog with graph databases and also works with relational databases. RuleML.org points out[74], "PSOA RuleML's databases (fact bases) generalize the instance level of Graph and Relational Databases; its knowledge bases complement facts by rules for deductive retrieval (extending the Datalog-level, function-free expressiveness of Deductive Databases to the Horn-logic expressiveness of Logic Programming), interoperation, and reasoning, as well as for optionally emulating part of the schema level." This seems to be stating that PSOA works with both relational and graph databases.
- **GQL/Cypher**: GQL[75] is an ISO project[76] to create a global standard query language (like SQL) for graph databases, graph query language. Open Cypher[77] which is based on Cypher which is the query language of Neo4j.
- **XBRL+SBRM+More**: XBRL[78] is an open standard technical syntax published by XBRL International, SBRM[79] is a forthcoming standard to be published by

---

[64] Shawn Riley, *Modern Symbolic AI in 2020*, https://medium.com/@shawn.p.riley/modern-symbolic-ai-in-2020-dfcc27abbc5c
[65] SWI Prolog, https://www.swi-prolog.org/
[66] Scryer Prolog, https://github.com/mthom/scryer-prolog
[67] Wikipedia, *Comparison of Prolog Implementations*, https://en.wikipedia.org/wiki/Comparison_of_Prolog_implementations
[68] Wikipedia, *Horn Logic*, https://en.wikipedia.org/wiki/Horn_clause
[69] ISO, *ISO Prolog*, https://www.iso.org/standard/21413.html
[70] Wikipedia, *Datalog*, https://en.wikipedia.org/wiki/Datalog
[71] RuleML.org, http://ruleml.org/papers/Primer/RuleMLPrimer2012-08-09/RuleMLPrimer-p3-2012-08-09.html
[72] Wikipedia, *Graph Database*, https://en.wikipedia.org/wiki/Graph_database
[73] RuleML.org, *PSOA*, http://wiki.ruleml.org/index.php/PSOA_RuleML
[74] RuleML.org, *PSOA RuleML Bridges Graph and Relational Databases*, https://wiki.ruleml.org/index.php/PSOA_RuleML_Bridges_Graph_and_Relational_Databases
[75] GQL Standards.org, *GQL Standard*, https://www.gqlstandards.org/
[76] Wikipedia, *GQL Graph Query Language*, https://en.wikipedia.org/wiki/GQL_Graph_Query_Language
[77] OpenCypher.org, *Open Cypher*, https://www.opencypher.org/

OMG that formalizes a logical conceptualization of a business report. While XBRL provides the functionality to represent all that is needed to express knowledge and much of what is necessary to process that knowledge and prove the knowledge is represented correctly. However, certain specific processing is missing that must be supplemented to create a complete system. As such, that additional processing logic must be provided.

There are undoubtedly other logics that can be used to process XBRL-based digital financial reports. Other completely different approaches such as the decision model approach[80] could possibly be used but would need to include an ontology-type component. Any syntax used should be 100% convertible to all other syntaxes and be able to round tripped back into the original syntax. Then, you could switch between whatever approach.

Converting between these logics is very possible. For example, converting between RDF and labeled property graphs is possible[81]. Converting from RDF to SWI Prolog is possible[82]. But 100% conversion is limited to the least common denominator, the set of logic that each alternative possesses.

1.12. Reconciliation of Terminology

Each implementation alternative tends to have its own unique jargon or terminology. The following is a reconciliation between the high-level terms used by each implementation approach:

SBRM Term	Prolog Term	Labeled Property Graph Term (Cypher/Neo4j)	Graph-theoretic (Graph theory) Term	RDF Graph Term (OWL, RDF, SHACL)	PSOA (RuleML) Term	XBRL Syntax Term	XBRL Abstract Model 2.0
Model	Knowledge base	Knowledge base	Directed Graph	Triple store; OWL ontology, SHACL rules, RDF graph	Knowledge base	XBRL instance + XBRL taxonomy	Document or Manifest
Structure	Compound Term	Node?	Vertex?		Compound Term	Network, Hypercube	Cube; Cube Region;
Statement	Expression; Clause				Expression; Clause		
Term	Term	Node	Vertex		Term	XML Schema Element	Model Element
Association	Predicate	Relationship	Edge		Predicate	Xlink arc	Aspect
Is-a	Predicate	Relationship	Edge label	SubClassOf	Predicate	Xlink arcrole; XBRL "general-special"; "wider-narrower"	Xlink arcrole; XBRL "general-special"; "wider-narrower"
Has-a	Predicate	Relationship	Edge label		Predicate	Xlink arcrole (mechanism exists but specifics undefined)	Xlink arcrole (mechanism exists but specifics undefined)
Property-of	Predicate	Property	Key-value pair	SubObjectPropertyOf	Predicate	Xlink arcrole; XML attribute	Xlink arcrole; XML attribute
Same-as	Predicate	Property	Key-value pair	EquivalentClasses	Predicate	Xlink arcrole; XBRL "essence-alias"	Xlink arcrole; XBRL "essence-alias"
Rule	Rule			SHACL constraint	Rule	XBRL definition relations; XBRL calculation relations; XBRL Formula	XBRL definition relations; XBRL calculation relations; XBRL Formula
Fact	Fact	Node?	Vertex?	Individual	Fact	Simple Fact	Data Point

[78] XBRL International, https://www.xbrl.org/
[79] OMG, SBRM, https://www.omg.org/intro/SBRM.pdf
[80] Wikipedia, *Decision Model*, https://en.wikipedia.org/wiki/Decision_model
[81] Neo4j, Jesús Barrasa, *RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?*, https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/
[82] Samuel Lampa, *SWI-Prolog as a Semantic Web Tool for semantic querying in Bioclipse: Integration and performance benchmarking*, https://www.diva-portal.org/smash/get/diva2:398839/FULLTEXT01.pdf

1.13. Other Possible Implementation Alternatives

The following are other possible implementation alternatives[83]:

Business Rules Processor	For more information	Approach	Information Syntax/Form at	XBRL Processor	Formula Processor	Support s Inferenc e	Problem Solving Method	Probabilis tic Reasonin g (Modal	State Machi ne	General Problem Solving Logic Capabilities	Fact Database	Knowledge Base	Rule Creation Interface (XBRL Taxonomy, XBRL Formula, Other)	g of Business Report Conceptual Model	g of Financial Report Conceptual	Explanation Mechanism (Transparency into Line of Reasoning and	Knowledge Acquisition Mechanism
Arelle (Open source API level interface)	http://arelle.org/	XBRL	Standard XBRL	Yes	Yes	No	Sequential	No	No	INCOMPLETE, Limited to XBRL Formula	XBRL Instance	XBRL Taxonomy	None provided	No	No	Not provided, can be created	Manual
XBRL Development Tools (Altova)	https://www.altova.com/xbrl-tools	XBRL	Standard XBRL	Yes	Yes	No	Sequential	No	No	INCOMPLETE, Limited to XBRL Formula	RaptorXML+XBRL Server	RaptorXML+XBRL Server	XBRL Specific but oriented to technical users	No	No	Not provided, can be created	Manual
Interstage Xwand (Fujitsu)	http://www.fujitsu.com/global/products/software/middleware/application-	XBRL	Standard XBRL	Yes	Yes	No	Unknown	No	No	INCOMPLETE, Limited to XBRL Formula	XBRL Instance	XBRL Taxonomy	XBRL Specific but oriented to technical users	No	No	Not provided, can be created	Manual
Sphinx (CoreFiling)	https://www.corefiling.com/products/sphinx/	XBRL	Standard XBRL	Yes	Yes	No	Sequential	No	No	INCOMPLETE, Limited to XBRL Formula	XBRL Instance	XBRL Taxonomy	XBRL Specific but oriented to technical users	No	No	Unknown	Manual
Clean Score (XBRL Cloud)	https://www.xbrlcloud.com/cleanscore.html	XBRL-based Business Reporting	Profile based Standard	Yes	Yes	Yes	Sequential	No	No	GOOD (Subset of RuleLog)	XML Infoset stored in file system	XML Infoset stored in file system	None provided	Yes	Yes	Good, usable by business professionals	Manual
Pesseract Knowledge Based Financial Report Creation System	http://pesseract.azurewebsites.net/	XBRL-based Business Reporting	Profile based Standard	Yes	No	Yes	Forward chaining	No	Yes	GOOD (Subset of RuleLog)	XML Infoset stored in file system	XML Infoset stored in file system	None provided at present time, will be business user oriented	Yes	Yes	Good, usable by business professionals	Manual
SWI PROLOG	https://www.swi-prolog.org/	Logic Programming	Open source defacto standard	No	No	Yes	Backward chaining (can do forward)	No	No	VERY GOOD (Turing machine; can be limited to DATALOG)	Proprietary or general format	Defacto standard PROLOG format	None provided	No	No	Unknown	Manual
CLIPS	http://www.clipsrules.net/	Logic Programming	Open source, based on PROLOG	No	No	Yes	Forward chaining	No	No	VERY GOOD (Turing machine; can be limited to DATALOG)	Proprietary or general format	Unknown	None provided	No	No	Unknown	Manual
FlexRule Business Logic Platform	http://www.flexrule.com/solution/	Business Rules	Proprietary or general format	No	No	Yes	Forward chaining	No	No	BETTER (Larger subset of RuleLog)	Proprietary or general format	Proprietary or general format	Comprehensive but oriented to technical users	No	No	Unknown	Manual
InRule (InRule Technologies)	http://www.inrule.com/products/inrule/	Business Rules	Proprietary or general format	No	No	Yes	Forward chaining	No	No	BETTER (Larger subset of RuleLog)	Proprietary format	Proprietary format	Oriented toward non-technical users and business professionals	No	No	Unknown	Manual
Smarts (Sparkling Logic)	https://www.sparklinglogic.com/smarts-decision-manager/	Business Rules	Proprietary or general format	No	No	Yes	Forward chaining	No	No	BETTER (Larger subset of RuleLog)	Proprietary or general format	Proprietary or general format	Comprehensive but oriented to technical users	No	No	Unknown	Manual
Fluent Editor (Cognitum)	http://www.cognitum.eu/semantics/FluentEditor/	Semantic Web Stack	Standard RDF, OWL, SWRL	No	No	Yes	Forward chaining	No	No	BETTER (Larger subset of RuleLog)	RDF stored in file system	RDF, OWL, SWRL stored in file system (Semantic	Comprehensive but oriented to technical users	No	No	Unknown	Manual
TopBraid Platform (TopQuadrant)	https://www.topquadrant.com/technology/topbraid-platform-overview/	Semantic Web Stack	Standard RDF, SHACL, RDFS, OWL	No	No	Yes	Forward and Backward	Yes	No	BEST (RuleLog plus)	RDF triple store repository	SHACL, RDFS, OWL, SPIN stored in file system (Semantic	Comprehensive but oriented to technical users; rule creation templates usable by	No	No	Good, usable by business professionals	Manual or Automated
Enterprise Data Governance (EDG); TopQuadrant	https://www.topquadrant.com/products/topbraid-enterprise-data-governance/	Semantic Web Stack															
Ergo Logic System (Coherent Knowledge)	http://coherentknowledge.com/financial-domain-application/	Semantic Web Stack	Standard RDF, RIF	No	No	Yes	Forward and Backward	Yes	No	BEST (RuleLog plus)	RDF stored in file system	RDF, OWL, RIF stored in file system (Semantic	Comprehensive but oriented to technical users	No	No	Good, electronic audit trail with provenance usable by business	Manual or Automated

Business rules management systems (BRMS)[84] and the decision model approach[85] tend to not use taxonomies/ontologies or high-level models. But an BRMS can store this information in other ways. Fundamentally, all rules processing is parsing some set of NAND logic gates[86].

So, any approach can effectively process information one way or another. The question is how ***efficiently*** can what you need to achieve actually be achieved. In the long run; the **cost of creating rules**, the **cost of managing rules**, and the **cost of maintenance** will far exceed the cost of a rules engine. Evaluating implementation alternatives needs to involve the total cost of ownership over the long term.

1.14. Word about "Rolling your Own" Solution

Another approach is to "roll your own" solution for processing the logic represented within an XBRL-based digital financial report. There are advantages and disadvantages to rolling your own logic engine. On the one hand, it can do precisely want you want, you are unconstrained by a more general implementation. Because your implementation would be specific rather than general, it would very likely be easier to use than a more general solution.

However, creating a rules engine or logic processing engine is a non-trivial task. Two software developers have created their own logic engines to process the specific

---

[83] *Rules Engines Comparison*, http://xbrlsite.azurewebsites.net/2020/library/RulesEngineComparison.jpg

[84] Wikipedia, Business Rules Management Systems, https://en.wikipedia.org/wiki/Business_rule_management_system

[85] Wikipedia, Decision Model, https://en.wikipedia.org/wiki/Decision_model

[86] Wikipedia, *NAND Gate*, https://en.wikipedia.org/wiki/NAND_gate

rules required by my method. Those two are XBRL Cloud and Pesseract[87] which is a working proof of concept.

The down side that I see with XBRL Cloud and Pesseract is the ability to add the next layer of logic processing, then the next, then the next. Software provides for literally an infinite ability to process more, and more, and more. A logic processing engine would need to continue to expand in order to keep up with the new ideas that can simply be handled "out of the box" with a powerful logic/rules engine.

1.15. Word about Enhanced XBRL Formula

Another potential implementation approach is to enhance XBRL Formula to fill in the gaps between XBRL Formula's current capabilities and the required capabilities that I have outlined.

1.16. Difference Between Knowledge Graph, Labeled Property Graph, and Logic Programming Language

In order to understand the differences between a knowledge graph (i.e. RDF) a labeled property graph (i.e. Cypher/GQL), and Prolog there are a number of useful papers and other helpful documentation:

- *RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?*[88] (by Neo4j which sells its labeled property graph)
- *KNOWLEDGE GRAPHS VERSUS PROPERTY GRAPHS*[89] (by Kristi Lee-John who works for TopQuadrant which sells knowledge graphs)
- *A graph DB vs a Prolog (or miniKanren)*[90], contrasts labeled property graph database to Prolog
- *Prolog-based Infrastructure for RDF: Scalability and Performance*[91]
- *An Introduction to Prolog and RDF*[92]
- *Limits of SPARQL: Introduction to Property Graphs*[93] (explains the limitations of SPARQL)

---

[87] *Guide to Building an Expert System for Creating Financial Reports*, http://xbrlsite.azurewebsites.net/2018/Library/GuideToBuildingAnExpertSystemForCreatingFinancialReports.pdf

[88] Jesús Barrasa, Field Engineer, Neo4j, *RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?*, https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/

[89] Kristi Lee-John, *KNOWLEDGE GRAPHS VERSUS PROPERTY GRAPHS*, https://www.topquadrant.com/download/knowledge-graphs-versus-property-graphs/

[90] A graph DB vs a Prolog (or miniKanren), https://stackoverflow.com/questions/29192927/a-graph-db-vs-a-prolog-or-minikanren

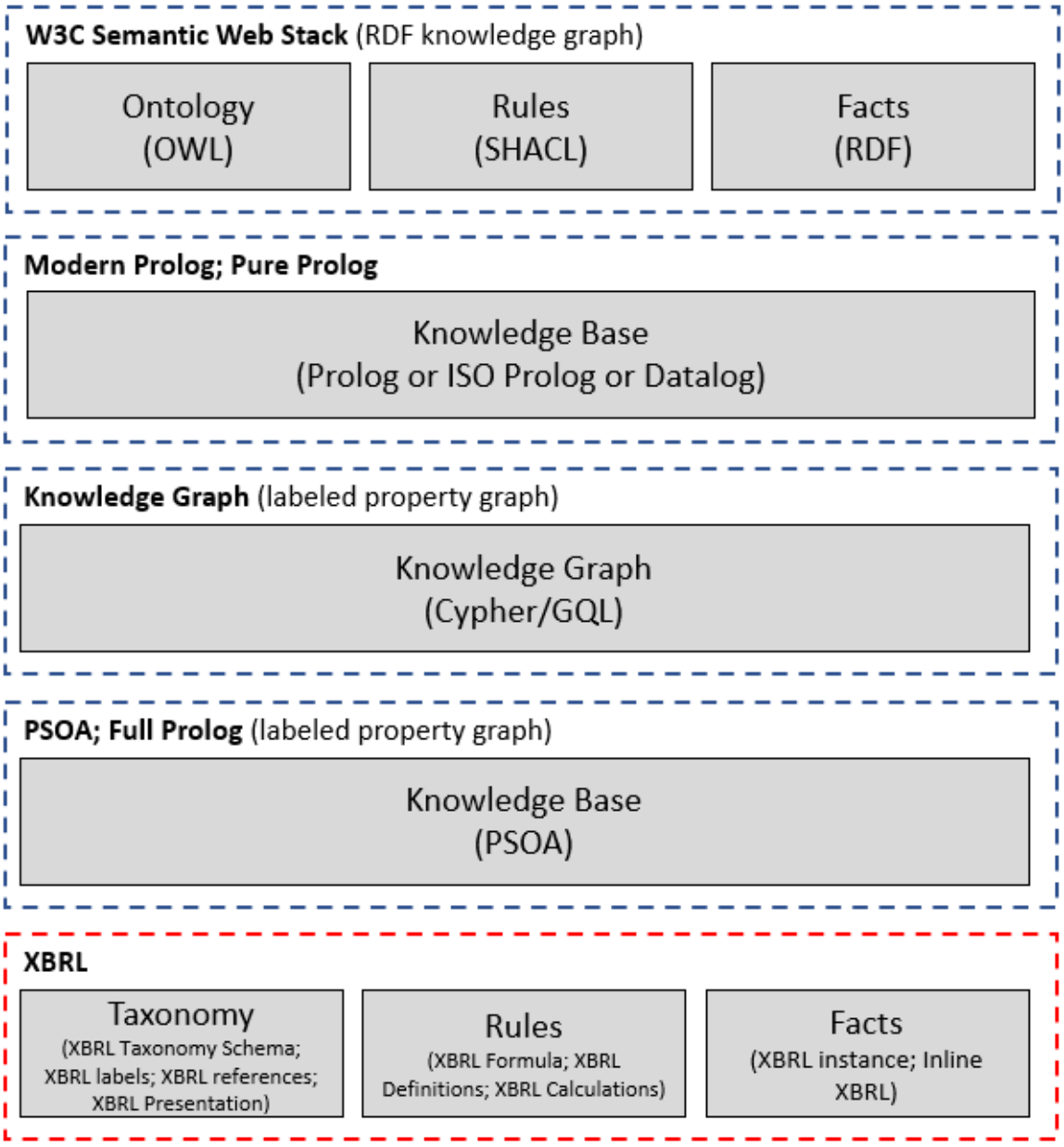
[91] Jan Wielemaker, Guus Schreiber, Bob Wielinga, *Prolog-based Infrastructure for RDF: Scalability and Performance,* https://km.aifb.kit.edu/ws/psss03/proceedings/iswc-03_1.pdf

[92] Bijan Parsia, *An Introduction to Prolog and RDF*, https://www.xml.com/pub/a/2001/04/25/prologrdf/

[93] Dr. Markus Krötzsch, *Limits of SPARQL: Introduction to Property Graphs*, https://iccl.inf.tu-dresden.de/w/images/4/47/KG2018-Lecture-09-overlay.pdf

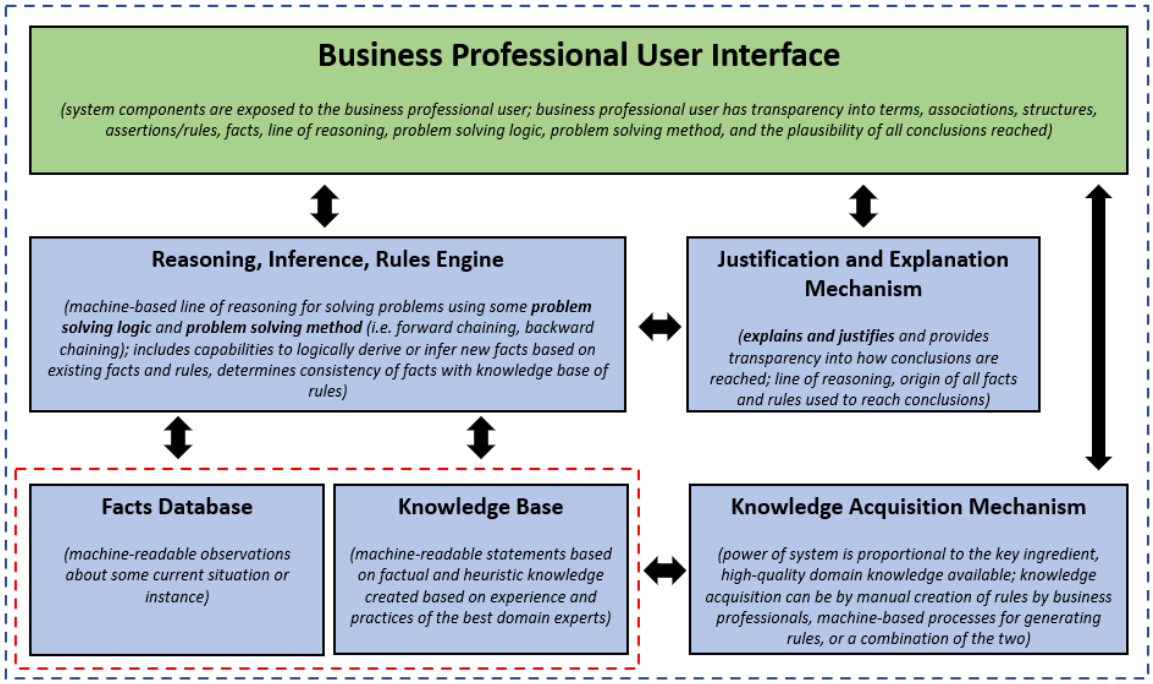
Each alternative has a set of pros and a set of cons. All three implementation approaches seem sound. I am not going to venture into which approach is the "best" approach.

What is clear is that it is very possible to convert (round trip) between RDF-based knowledge graphs, labeled property graphs, Prolog, and XBRL technical syntax formats.



1.17. Knowledge Based System Components Required

The following graphic shows the components of a knowledge based system:



I want to focus for a minute on the "Fact Database" component, the "Knowledge Base" component, and the "Reasoning, Inference, Rules Engine" component.

A "Reasoning, Inference, Rules Engine" needs to perform the following tasks and information needs to be within the "Fact Database" and "Knowledge Base" to enable those tasks:

- **Conformance of Model to Meta-model**: A model that is created (i.e. a report) needs to verified against a specified meta-model (i.e. such as SBRM, EDGAR Filer Manual for SEC filings, ESEF for ESMA filings, etc.) to be sure the model is consistent with that meta-model.
- **Type-subtype construction inconsistencies**: Types and subtypes defined in and/or used by a model need to be consistent with what is expected. For example, a concept "Cash and cash equivalents" that is defined as a type of "Current Asset" must only be represented as a "Current Asset" and not erroneously, for example, as a "Noncurrent Asset".
- **Factual contradictions and inconsistencies**: Facts reported within a model (i.e. statements made) must not contradict or be inconsistent with other facts reported or statements made. For example, if a rule is defined (i.e. a statement made) that "Assets = Liabilities + Equity" and Assets is reported to be $5,000, Liabilities is reported to be $4,000, and Equity is reported to be $4,000; then the rules engine must report that inconsistency/contradiction in statements.
- **Mathematical inconsistencies**: Facts reported within a model (i.e. statements made) must be consistent with mathematical rules specified at the model level or at the structure level to be sure that such mathematical relations are consistent with what has been specified.
- **Structural inconsistencies**: If a structure is defined, say "Inventory Components Roll Up" and statements are made that indicate that that

structure should be a "Roll Up" and that the total concept of that roll up must be the concept "Inventories, Net"; then if the structure does not have that characteristic, that inconsistency must be reported to a user. Further, if a structure is specified to be a "Roll Up" then a rule (i.e. statement) must be provided that specifies the roll up mathematical relations of that structure (i.e. the rule cannot be missing from the model).

- **Reporting inconsistencies**: If a structure is specified to be required to be reported within a model and that structure is not provided; then a user of that model must be informed of that inconsistency. For example, if there is a statement made that a "Balance sheet" is a required structure to be provided within a model; if that structure does not exist then the user of the model should be informed of this inconsistency which was discovered.
- **Sanctioned inferences derived**: A "sanctioned" or "authorized" inference is defined as an inference deemed as an appropriate conclusion to draw from the explicitly provided information provided by a model. A rules engine must be able to derive all sanctioned inferences. For example, if a report provides information about "Assets" and about "Current Assets" and provides a rule (i.e. statement) specifying that "Assets = Current Assets + Noncurrent Assets"; then the rules engine must be able to compute the value for "Noncurrent Assets" based on the other information provided.

And so, the "Fact database" and "Knowledge Base" must hold information necessary to enable verification of the seven categories of tasks described above. Further, the "Reasoning, Inference, Rules Engine" must have the logical capabilities to use the provided information held in the "Fact Database" and "Knowledge Base" and to reach the conclusions expected by those seven categories.

And so, while there are different alternatives of achieving these objectives and each different alternative can be more efficient or less efficient at achieving these objectives; any alternative that is not capable of ***effectively*** satisfying the objectives is really not an acceptable alternative.

For example, type-subtype relations can be represented quite effectively using a taxonomy or ontology (i.e. set of statements that forms what amounts to a hierarchy of information). A semantic reasoner is quite efficient and effective at reading the taxonomy/ontology information and determining if a model is consistent with the type-subtype specifications as to what is and what is not permissible.

Business rules management systems (BRMS)[94] and the decision model approach[95] tend to not use taxonomies or ontologies to store information used by such a system's rules. But an BRMS can sore this information in other ways. That approach may, or may not, be as efficient as storing and processing information using a taxonomy or ontology; but it can work effectively.

Further, ontologies cannot represent mathematical computations but they can represent type-subtype and other relations effectively. On the other hand, BRMS systems can represent mathematical computations but do not leverage ontologies or models.

[94] Wikipedia, Business Rules Management Systems, https://en.wikipedia.org/wiki/Business_rule_management_system
[95] Wikipedia, Decision Model, https://en.wikipedia.org/wiki/Decision_model

The point here is that whatever approach is used, one needs to weave together all the functionality necessary to effectively process 100% of the required capabilities (i.e. the seven categories above) and effectively provide the required information.

2. Further Reading

For more information, please see:

- **Mastering XBRL-based Digital Financial Reporting**[96]
- **Logical Systems**[97]

[96] Mastering XBRL-based Digital Financial Reporting, http://xbrl.squarespace.com/mastering-xbrl/
[97] Logical Systems, http://www.xbrlsite.com/mastering/Part02_Chapter05.A_LogicalSystems.pdf