The Minds Eye™ semantic engine from Oracle

Table of contents

Architecture overview

For the purpose of solving many semantic data management problems, Oracle's strong support for industry standard semantic representations (and their manipulations) is a necessary addition to the Oracle database. However, standard semantic representations are not sufficient for solving problems that require a semantic extraction or interpretation process in order to merge[1] so-called "**unstructured**" input data with the "**structured**" data already residing in the database. This is especially true in the field of medical imaging.

In order to better understand the problem that is solved by the Minds Eye™ Oracle semantic engine, it will be helpful to introduce a few concepts and clarify a number of misleading industry standard terms.

Consider first the pair of industry-standard terms "unstructured" and "structured". The implication that there is more structure in structured data than in unstructured data is untrue.

A better way to think about the distinction implied by the use of these two words is in terms of the relationship between

- the structures explicitly represented (and thus queryable) at the time the information is captured and
- the structures in terms of which the information's users would like to query.

If the query structures are the same as or a subset of the capture structures, the information is explicitly structured (or simply "structured"- in industry terms).

For example, classic relational and so-called semantic data are both captured and queried in terms of tables and fields, and triples. Such information is thus "explicitly structured". In contrast, a scan of an X-ray captures only grey scale values in an XY grid. If all that users wanted to know was the grey scale value at some combination of XY locations, the image data would be explicitly structured for that purpose. But this is not what people want to know. They want to know whose X-ray is it? Was it diagnostic of anything? Who ordered the X-ray? Where was it taken? Was this image a human-detectable duplicate scan (quite different at a pixel level) of any other image in the system? Or was it perhaps a significantly different near duplicate in the sense that it covers the same anatomical region as does another image for the same person except that this new image shows a tumor where none had been before?

Not only are scans of medical records implicitly structured, but they are frequently "multi-structured" as well. Information is multi-structured when a single unit of data

[1] The term "merge" is more appropriate than "update" because new information may link in numerous ways to the combination of data and definitions that exist in the database.

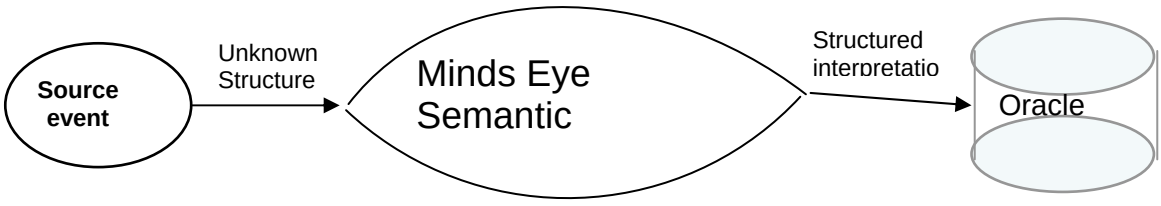
input such as a file may contain multiple different kinds of information each requiring a distinct semantic interpretation process. For example, a scan of an X-ray that contains textual information such as the name of the patient, the name of the hospital and a date, would be an example of multi-structured information. The X-ray image is one kind of structure requiring one kind of interpretation process; the embedded text is another. If there were a handwritten note on the image, that would be yet another. . A spreadsheet where clues about the meanings of terms were indicated by fonts would be another example. Since there are no key words or headers or meta tags telling the semantic engine how to interpret the information, the engine must discover how to do so on its own.

From the perspective of the ever growing catalog or library of what is known by the semantic engine, information and thus required representations for that engine need to be "**multi-modal"**. This means that the same piece of understanding, such as a diagnostic condition for a person, may be physically represented in a number of different ways: an MRI, an X-ray, a doctor's report. Even just words alone as physical representations, may constitute different physical representations: be they different terms such as myocardial infraction vs heart attack or different languages such as crise cardiaque in French.

Ultimately, an entry in the library/catalog such as "at risk of stroke" may be inferred from the presence of multiple physical representations such as blood reports, tissue samples, antigen presence and MRI. In other words, it may be a combination of multiple physical representations realized over time and space that signals the presence of a single more abstract condition or event.

Thus, from a high level, as illustrated in figure 1 below, the Minds EyeTM semantic engine from Oracle transforms implicitly multi-structured information via a multi level multi-modal library into explicitly structured information capable of being queried, manipulated and managed through understood structured interfaces such as SQL, and of being merged into the Oracle DB.

Figure 1. High level view of the Minds Eye semantic engine from Oracle



We would call any software capable of performing the appropriate transformations a semantic engine. It should also be noted that any software that warrants being called a semantic engine also needs to support logical inferences and computations. The process of semantic interpretation depends on a huge amount of inferencing and computational

brawn. In other words, to be a semantic engine is to be a logical-semantic-computational engine.

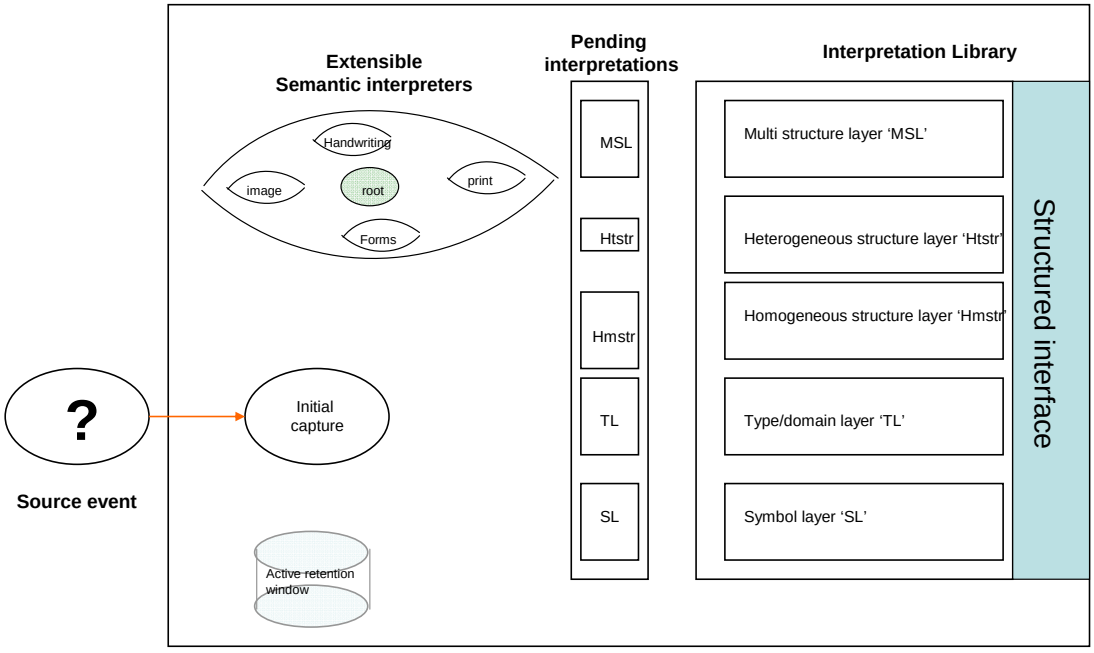
For any new piece of information, the major findings of the semantic engine include:

- The new piece of information is an effective duplicate of something already in the database. When this is the case, the engine will further attempt to determine which duplicate is of higher quality
- The new piece of information is a new form for something already known such as a new written diagnoses or a new diagnostic images for a known condition
- The new piece of information contains new attributes for individuals already known such as new diagnoses or new address information for a known individual
- The new piece of information is a known attribute such as a diagnostic image for a new – unknown- individual
- The new piece of information has no discernible structure beyond its capture structure

Let's drill down now to gain a better understanding of how that occurs. As illustrated in figure 2 The Minds EyeTM Oracle semantic engine consists of six major components:

1. An initial capture zone
2. An active retention window
3. An extensible semantic interpreter
4. A pending interpretation zone
5. An interpretation library and a
6. Structured interface

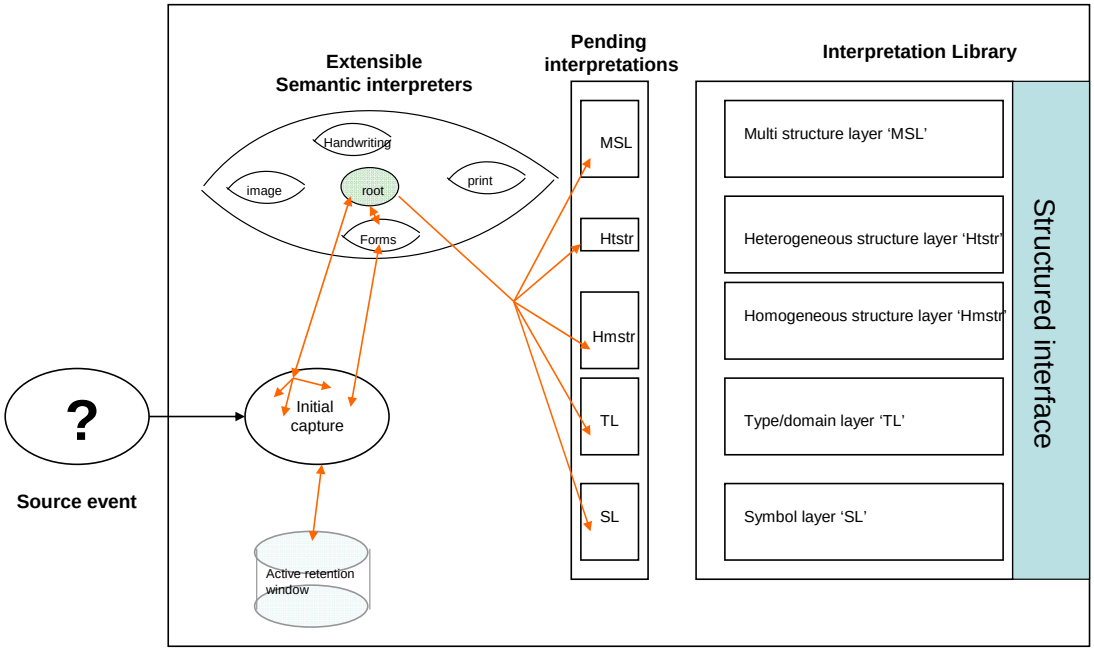
Figure 2 Overview of the Minds Eye semantic engine from Oracle



As illustrated in figure 3 below, every time something new and unexpected enters the system, the root interpreter goes through a set routine to figure out the kind(s) of information that may be contained. Depending on the outcome of that process, any number of specific semantic interpreters such as forms, images and/or text may be invoked independently or in unison; and on any subset of the initial information..

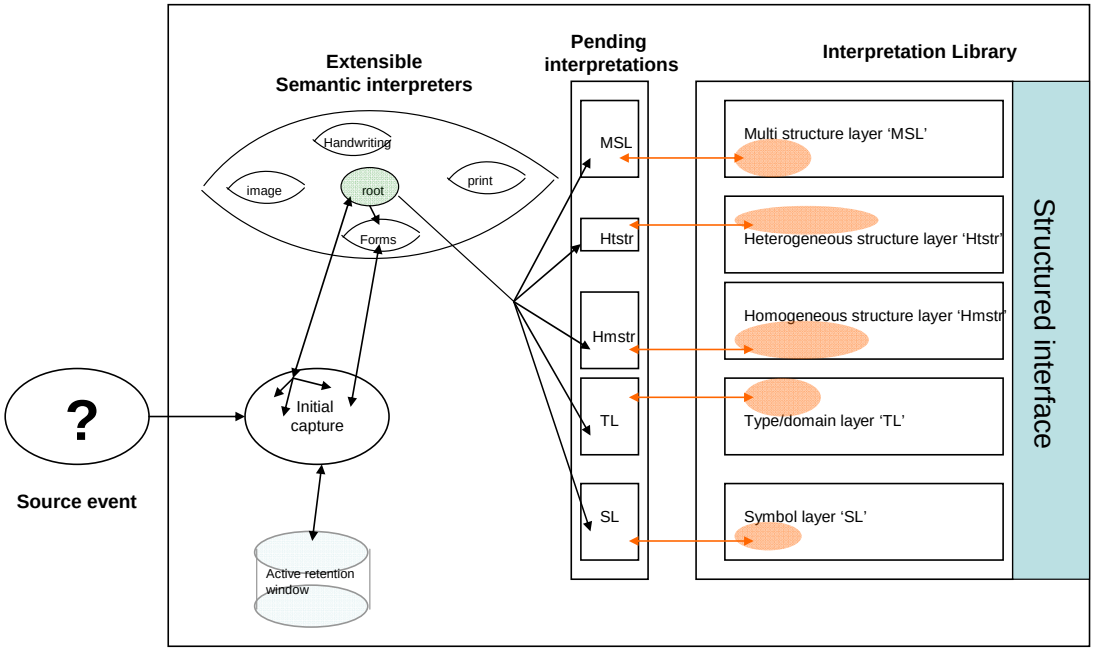
Each structure-specific interpreter works across a number of different levels of interpretation and receives feedback from the library where appropriate. The process is geared to work with partial information. The final product of each of these semantic interpreters is a tentative interpretation.

Figure 3 Overview of the Minds Eye semantic engine from Oracle



As shown below in figure 4, predominantly bottom-up interpretations from the semantic interpreters interact in the pending interpretation zone with the predominantly top-down interpretations from the library to form a final interpretation that is then merged into the library. Links are retained that associate the library definition(s) with the locations of the relevant source events.

Figure 4 Overview of the Minds Eye semantic engine from Oracle



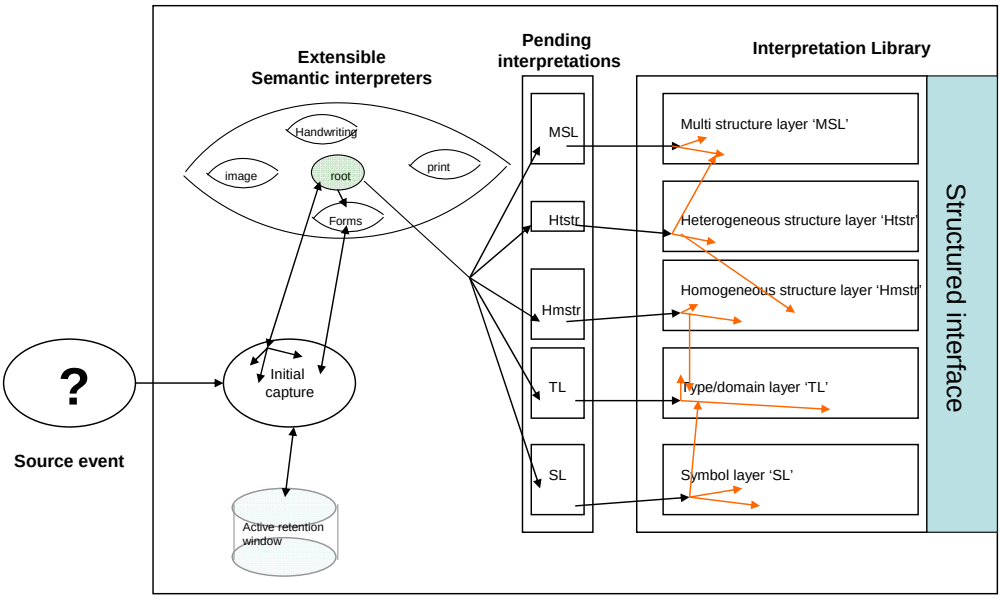
As shown in figure 5 below, the merging of new interpretations into the library may set off any of a number of conditioned responses from alerting an analyst to the possible presence of a duplicate to the discovery that an image appears to belong to a specific serviceman or that an image scanned in as if belonging to a particular service person most likely belongs to someone else.

The library encodes all logical structures and their probabilistic positional and resolutional linkages from single values to the largest application that has been seen and is capable of being recognized.

The main logical/semantic library levels are

- Type
- Homogeneous Structure
- Heterogeneous structure
- Multi-structure

Figure 5 Overview of the Minds Eye semantic engine from Oracle



The library manages so-called language objects, such as spreadsheet references, in the same fashion as so-called physical objects such as image patterns.

For example, in the same way that the system may recognize a physical scene to be

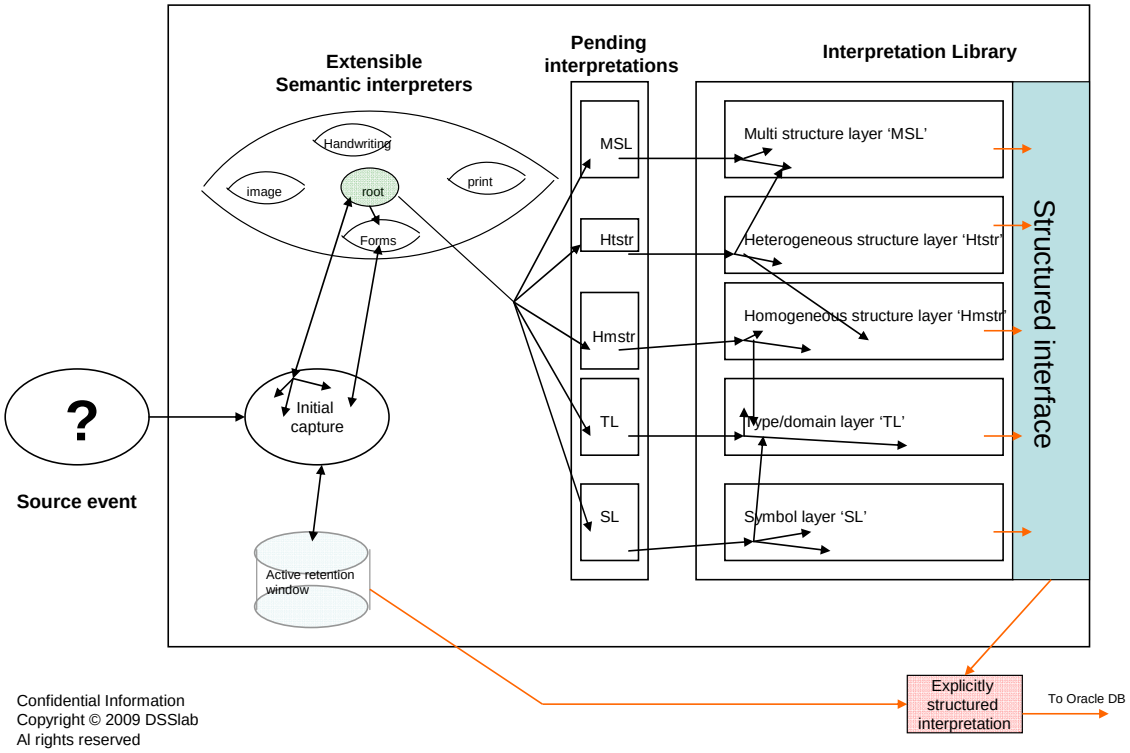
(
instance *square* of type “Closed segment series” joined under to
instance *triangle* of type “Closed segment series”
) adjacent to
instance *circle* of type “Closed segment series”
=>
instance “*house and sun”* of structure “group of closed segment series”

The system could recognize a language scene to be

(
instance *January* of type “Time” joined intra locator tuple with
instance *Cambridge* of type “Store”
) joined 1-1 with
Content Instance 500 of type “Sales” unit “Dollars”
=>
instance “*January, Cambridge, 500 dollars”* of structure named “Sales Model”

Finally, as shown in figure 6, below, new interpretations (with or without their triggered consequences), are exported as structured data to the Oracle database. Where appropriate, and by leveraging the structured interface users, of the semantic engine can directly query and explore the data as it is being interpreted by the semantic engine.

Figure 6 Overview of the Minds Eye semantic engine from Oracle



Logical foundations of the Minds Eye™ semantic engine from Oracle

The Minds Eye semantic engine from Oracle is grounded in a highly advanced typing system developed over a twenty five year period and stress tested in many world class domains from international financial institutions, enterprise information systems, and seismic analysis to integrated socio-economic-environmental modeling.

Provisional patents have been filed and utilities are in the process of being filed to cover all novel aspects of the Minds Eye semantic engine including:

- Semantic understanding of spreadsheets
- Multi-modal "object" recognition
- Term extraction from NL text
- Segment extraction from images
- Object, surface and segment extraction from tactile sensations

Introduction

The power of both the Minds Eye library and the Minds Eye semantic interpreter follow directly from the power of the underlying typing system which provides all the capabilities of logic, language and mathematics (and obviating their known inconsistencies) within a single compact extensible system.

The major constructs of the typing system are that of

- Types (or Domains) as collections of instances of some unit or collection of related units (where units themselves are a role played by a type), with orderings and potential operators associated with those values,
- Schemas as particular collections of functionally related Types capable of supporting the definition and execution of expressions.
- Expressions as particular collections of inter-related types that exist relative to some implicit or explicitly defined schema in both an exchanged and an executable form.

For example, the popular notions of number system, dimension, hierarchy, measure, attribute, variable, data type, network, directed graph, (possible subject and possible predicate in the natural language sense of these terms), and (possible function and possible argument in the predicate logical sense of these terms), may be thought of as specializations of the more general notion of Type.

Furthermore, for example, the popular notions of model, (world -as in possible worlds), multidimensional hypercube, multidimensional multi-cube, Relation, Class diagram, frame, script, system of equations, shape file, process, application and program may be thought of as specializations of the more general notion of Schema.

Within the Minds Eye typing system, expressions are (the thing that, or), what does things relative to types and schemas. For example, expressions

- Assert and question propositions
- Specify and execute calculations
- Modify schemas
- Create new schemas
- Modify non-primitive Types
- Question any aspect of a type
- Create new non-primitive Types and
- Create new primitive Types

The Type system primitives are composed of "constructs", "ordering relations" and "operators".

Since operators are constrained by the constructs and orderings within the types to which they apply, and since all expressions are defined in terms of types (or other expressions which at some point are defined in terms of types), the limits of intelligible expressions are determined by the type system in place at the time of the existence of the expression. Of course, those limits may vary over time and between spatially differentiated systems.

Seemingly higher level concepts like trust, logical state and want are naturally defined in terms of second order expressions i.e., expressions that take other expressions as arguments and function as subsystems within the overall MINDS EYE System.

Finally, all explicitly cognitive processes or competencies are defined in terms of systems of schemas wherein schemas may exist in any kind of relation with other schemas (including M-to-N), and wherein expressions in some schemas may query, edit, activate or deactivate other schemas.

Types

Types are akin to a generalized form of number system[2]. They include some notion of valid or potential values, some notion of valid operations and some other stuff that will be described below. Suffice it to say that any understanding of types begins with an understanding of their composition or primitive components.

Before launching into a detailed description of the MINDS EYE System's primitive type components, or simply primitives, it is important to understand how these primitives relate to each other.

[2] Depending on the reader's background, the closest points of reference are Type Theory or Category Theory

The three type components, namely constructs, ordering relationships and operators are each projections of a single underlying unity. The token-based separateness of such seemingly distinct notions as, for example, “word”, “addition”, and “hierarchy” masks a deeper unity. The purpose of the following section is to point towards that unity.

Constructs

The major constructs in any model are the primitive linguistic objects that enter into relations with other linguistic objects. All expressions made within a model are in terms of the constructs of that model. For OLAP models, the major constructs include that of Dimension, Measure, attribute, cube and instance. In the MINDS EYE System the major constructs include the notion of Type and Type Structure. And within Types there are the notions of unit and value. Although it is possible to write or say the term “Type” or “Type Structure” or “Unit” or “Value” independent of uttering any other term, it is not possible to define any one term absent referring to some other term. For example, any definition of the term “Type” would need to make reference to the terms “Unit” and “Value”.

One of the basic characteristics of a Type is that it must define some set of potentially usable values, called potential values. A “Sales” Type might be defined as a dollar value greater than or equal to zero. A “Product” Type might be defined as any 8 Byte Char. Regardless of what kind of Type is defined, there is an implicit assumption that the values of the Type are distinguishable. The Sales value “$100” is not the same as the Sales value “$150”. The Product value “Shoes” is not the same as the Product value “Hats”. If values could not be distinguished, Type definitions would be impossible. Yet, the notion of value comparisons and of equality, is not itself a construct. Rather the notion of value comparisons is a primitive operator or function. Thus, the specification of primitive constructs presupposes a well defined notion of primitive operators.

Another basic characteristic of a Type is that it must be possible to traverse its potential values. There need not be a sequential ordering as there is for numeric series, but there must be some way to move from value to value. If the potential values of a Type could not be traversed, it would be impossible to call a function that changed the value of a Type based on some condition. For example, the ability to add $5 to the current value of $20 of a Dollar Type that represents an hourly wage presupposes that the value $20 is connected to the other dollar values of the Type so that it can be incremented by $5 to yield $25. As such, the specification of primitive constructs presupposes a well defined notion of primitive ordering relationships.

Thus, the primitive constructs of a model/system presuppose specified ordering relationships, and functions in addition to other primitive constructs.

Ordering Relationships

Every Ordering Relationship presupposes both the constructs that are ordered, as well as the constructs used to specify the ordering relationship (typically integer values as in the expression "1 to N"). Furthermore, every Ordering Relationship is expressed in terms of a particular function, namely the function that traverses from one construct in the ordering relationship to another. In other words, in a hierarchical ordering expression of the form "1 to N" the word "to" masks an implicit function, namely that traversing from the "1" to the "N" requires a single step or unit delta in the down hierarchy direction. While traversing from any "N" to the "1" requires a single step or unit delta in the up hierarchy direction.

Thus the specification of Ordering Relationships presupposes the existence of both constructs and functions.

Operators

Consider an operator as seemingly primitive as comparative equality or "=[3]". We might say, for example, that Sales = Costs. But can we define the operator "=" absent referencing constructs and Ordering Relationships? Here again, we can not.

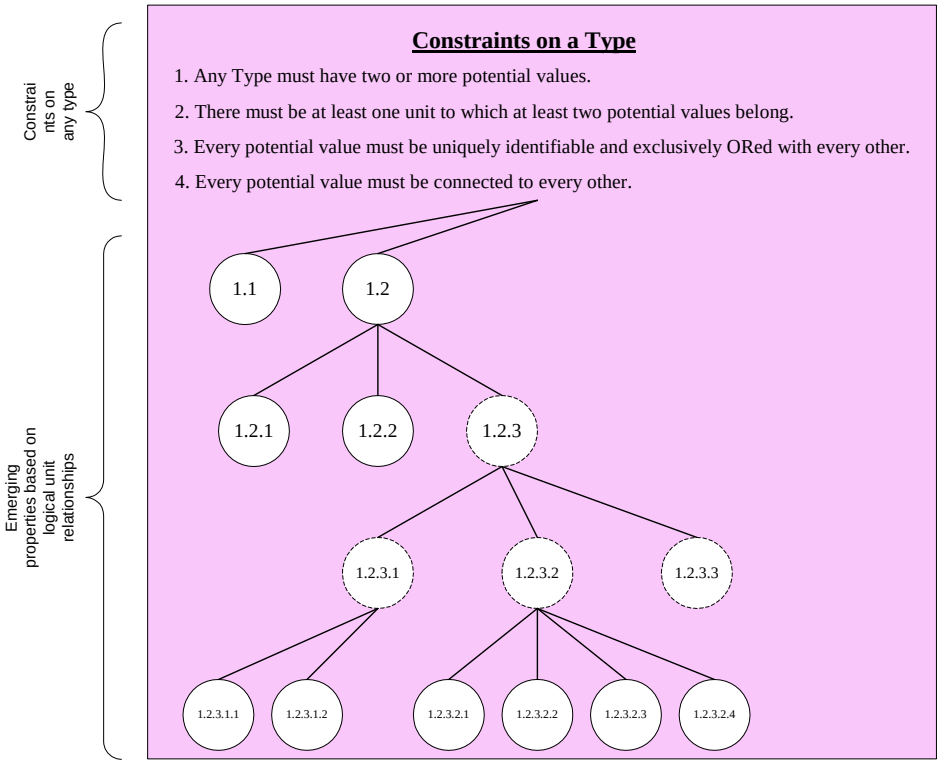
For example, any definition of the equality operator would need to reference those Constructs whose equality is being tested or asserted. In addition, there is an implicit Ordering Relationship in the equality operator, namely that of "1 to 1". In other words, and continuing with the example above, one value of Sales is equal to one value of Costs. If there were more or less than one value from each of the two Constructs, the operator would not work as defined.

Thus, the specification of Operators, like Ordering Relationships and Constructs, presupposes the existence of the other primitive concepts as a part of their very definition!

[3] Though in most programming languages "=" denotes assignment and "==" denotes comparison

The decision tree of emerging types

Treating the general constraints as the first node in a classification tree, we can describe the main kinds of types in terms of the following unit relationships.



- 1.1 If any of these constraints are not met by a Type, that Type is invalid
- 1.2 If all the constraints are met, and the Type has only one unit, the Type is a classic mono-level Type of the kind postulated in the Relational model and Date's more recent third manifesto.
- 1.2.1 If the Type has more than one unit and the units are not all connected, the Typeis invalid.
- 1.2.2 If the Type has only one unit and only two potential values, it is typically called a Boolean.
- 1.2.3 If the Type has more than one unit and the units are fully connected then there are a wide variety of valid Types that might be defined depending on the logical connectives thatjoin the units.
- 1.2.3.1 ANDed units
- 1.2.3.2 ORed units
- 1.2.3.3 AND/OR mixture
- 1.2.3.1.1 Specifically, if the units are all ANDed together, and there exists some functional relationship betweeen the units as there does, for example between distance, time an speed, the Type and whatever units(s) is(are) deemed dependent are compound.
- 1.2.3.1.2 If the units are ANDed together and there does not exist any functional relationship between the units as there does not, for example between a product name and SKU, the Type and the collection of units are considered to be concatenated.
- 1.2.3.2.1 If the units are all ORed together and they translation function between them defines a partial ordering on the set of units, the set of units forms a hierarchy.
- 1.2.3.2.2 If the units are all ORed together and the translation function between them defines a single unit relative to which all other units are defined, the set of units forms a hub and spoke unit set.
- 1.2.3.2.3 If the units are all ORed together and the translation function between them defines a set of network-style connections the set of units forms a peer-to-peer unit set.
- 1.2.3.2.4 If the units are all ORed together and the translation function between them defines a set of hierarchical units nesterd within a peer- to- peer unit set, the Type is considered to have a unit set. The unit relationships between the British system of pound weights and the Metric system of weights has this form.

KEY

Novel attributes of the Minds Eye™ typing System

Ontology-free representations

The notion of Type as an underlying collection of possible values obviates the need for the ontological commitments and associated modeling conundrums that affect ER, OO, RDF/OWL, and other approaches whose primitive modeling constructs are intended to represent certain things in the world. Rather, Types are a model of the language requirements for representing the world, not a model of the world. As such, nothing is claimed to be a Type.

Processes can be modeled as Types in the same way as objects. The process steps are captured as a location structure to which arbitrary measures may apply as contents. The notion of ordinal Time is built in to the process Type. Calculations of relative process concurrency are directly supported. Processes can spawn sub-processes and can be rolled up into higher level process abstractions. In short, processes and objects are just different specializations of type. sophisticated process modeling is directly supported in the MINDS EYE Model.

Types can be as easily used to define a (complex) data source in terms of a large collection of simple Types between which many complex relationships exist, or in terms of a smaller number of more complex Types between which a smaller number of simpler relationships exist. The line between intra-Type modeling and inter-Type modeling is dependent on the subjective determination of where to define linguistic surfaces. This representational flexibility is crucial because the same objective reality or set of data sources can be legitimately interpreted in any number of ways

Since Type Structures can easily embed in other Type Structures, the MINDS EYE Model directly supports nested and dynamic schemas.

Since the definition of a Type or Type Structure can vary between uses and since equivalency relationships can be established between any Types that share at least one primitive Type, the Minds Eye typing system directly supports the ability for multiple users to collaborate by dynamically linking their personal views with a common underlying model or by linking their models.

The combination of flexible representation, general equivalency mappings and nested ordering relationships supports very powerful and general semantic mapping.

The combination of ordering relationships and representations supports representational intelligence. See the appendix on “the Application of Type language to the Semantic Description of visual forms”.

No unnecessary hardwiring of structural distinctions

Not only does the Minds Eye typing system eliminate the artificial boundaries between notions of object and process, it also eliminates a number of other arbitrary structural distinctions whose hard-coded embedding within information systems has hindered the ability of those systems to tackle complex recognition problems.

It is interesting to note that there is a tendency to treat general distinctions as referential or objectively classifying. This was true for Aristotle with his distinctions between substance and property as well as subject and predicate in his "Categories of Interpretation". This was true for Frege in the "Begriffshrift" with his logical distinction between function and argument. And this has certainly been true in the software industry with such distinctions as

- Data/metadata
- Context/object
- Dimension/measure
- Class/attribute/association

The problem is that these distinctions are not referential but rather functional, or use-specific. For example, a piece of information that may serve as so-called metadata in one context, say the information's author or time of creation, may serve as data for a different process such as a study of who has created what. Information such as time may serve as an organizing dimension in one context and a variable in another.

Solves significant problems in the foundations of logic, mathematics and language

- Provides a mechanical grounding for logic:
  - propositional, predicate (other, semantically enriched logics such as mereology, temporal logic, modal logic)
    - Does not suffer from paradox: Liar, Grellings, set of all sets,
    - Exhaustive, mechanically implemented method for processing all kinds of ill-formed expressions without resorting to multi-valued logics and the operational ambiguities they entail
      - Provides a mechanical definition of well-formedness that exactly specifies the conditions for meaning

- Provides a mechanical grounding for mathematical number systems:
  - Naturals, Integers, Rationals, Reals, Complex, Vectors, Matrices, Tensors, Directed Graphs, Networks

- Provides novel, consistent accountings of
  - Irrationals,
  - so-called “infinite sets”,
  - the definition of “number” itself,
  - the definition of “unit”
- Provides a novel consistent accounting of the relationship between Euclidian and non-Euclidian spaces
  - In terms of compound unit/metrics
  - Showing certain number system problems to be rather coordinate system differences

- Provides a mechanical grounding for language:
  - Human, other animal, machine
    - Replaces “part-of-speech” taxonomy for parsing human language with “type role”
      - Thus integrating deep linguistic structure and knowledge/understanding
      - Provides a mechanical definition of well-formedness that exactly specifies the conditions for meaning
      - Distinguishes the constraints for well-formedness between exchanged forms and compiled forms of expressions
    - Accounts for what is known of other species languages and machine languages
    - Provides a mechanism for creating the sensory-motor schema that may get used for parsing symbolic expressions

Guide to appendices

There are six appendices to this document.

- Appendix one is a condensed/truncated but detailed version of the typing system upon which the Minds Eye semantic engine is grounded
- Appendix two is an application of the typing system to the semantic characterization of visual forms used for analytical visualization. It has served as the basis for automating the selection of visual displays so that the semantics of the display match the semantics of the query. The material is excerpted from chapter 9 of OLAP Solutions 2nd edition by Erik Thomsen.
- Appendix three is a comparison of the Minds Eye typing system with the foundations of canonical mathematics
- Appendix four is a comparison of the Minds Eye typing system with the foundations of canonical logic
- Appendix five is a comparison of the Minds Eye typing system with canonical logic as relates to material implication across multi leveled information spaces
- Appendix six is an illustration of how the Minds Eye typing system escapes the set of all sets paradox

Please note that the technical name for the typing system is the LC model, a term that will be found repeatedly in the appendices. The abbreviation stands for Located Contents and refers to the Tractarian (as in the Tractatus-Logico-Philosophicus by Ludwig Wittgenstein) -inspired functional as opposed to referential approach to foundational issues.

