

# Semiotic Knowledge Models for Personal Knowledge Repositories

Stefano Casadei<sup>1</sup> <sup>a</sup>

Memorilog, Cambridge, MA, U.S.A.

**Keywords:** Knowledge Representation, Personal Knowledge Representation, Semiotic Model.

**Abstract:** Knowledge graphs have been used successfully to represent and acquire general knowledge and have also been proposed for personal knowledge representations. While general knowledge data can be modelled statistically as being a noisy projection of universal (and crisp) entities, categories, and relationships, personal knowledge data requires a more refined model: each user's peculiarities and fluctuations in associating words with meanings and meanings with words should be tracked and analysed instead of being treated as noise and averaged out. This position paper describes a semiotic knowledge model whose primitives are the signification events which occur when symbols such as words and linguistic expressions are associated with an instantaneous meaning. Semiotic structures constructed from these primitives with users' active participation, enable them to create, update, modify, organize, re-organize and curate detailed and comprehensive representations of their own personal knowledge by means of their own personal terminologies, taxonomies, and organizational schemes.

## 1 INTRODUCTION


Computer-implemented representations of knowledge and information of different kinds are widespread and underly many of our everyday activities, from doing a search on the internet to booking an airline ticket. While technologies for creating repositories of general knowledge have advanced significantly during the last few years, the problem of representing and exploiting personal information lacks an equally successful solution.

Knowledge graphs have been proposed as a means for representing personal information (Balog, Mirza, & Skjaeveland, 2022; Montoya, & al. 2018) but, as it has been pointed out by (Balog & Kenter, 2019) the particular nature of personal information presents unique challenges still in search of definitive solutions: 1) Entities of personal interest are typically mentioned only a few times and information about them can be sparse. This makes it difficult to apply statistical and deep-learning methods commonly used for general knowledge. 2) Relations in a personal knowledge scenario may be short-lived and common relation extraction methods may not be applicable. 3) Users may be more inclined to use their own organization schemes based on “freely defined

semantic categories” rather than external ontologies designed by somebody else.

To address these issues and to develop friendly user interfaces for knowledge repositories, this position paper describes a knowledge model whose symbolic primitives are more refined than those used in knowledge graphs, making it possible to build more powerful and more flexible symbolic constructs.

The constituents of typical knowledge graphs (named entities, types, attributes, relationships, etc.) and of many other types of knowledge representations, are *crisp* and *universal*: their meanings are assumed to be well defined, *constant* over time and *invariant* across all users. *Data* is modeled as a *noisy* projection of these underlying universal crisp constituents and the goal of knowledge acquisition (harvesting) is traditionally viewed as one of removing this noise and of “cleaning” the data to recover the underlying constituents. Consequently, many knowledge acquisition methods are of a statistical nature and require large amount of data to counter the noise (Ilyas & Chu, 2019; Weikum, Dong, Razniewski, & Suchanek, 2021; Kejriwal, Knoblock, & Szekeley, 2021).

<sup>a</sup> <https://orcid.org/0009-0004-0094-8392>

It is well-known, however, that entities, attributes, relationships, and especially types and categories are not always crisp and universal. For example, users think more in terms of “natural kinds” than rigid categories (Russell, Norvig, & Davis, 2010). This is particularly important in a scenario where knowledge is produced by users describing what is important to them. Consider for example two users, Alia and Barouk, who maintain a list of their friends. Alia may want to include co-workers in the same category of friends, while Barouk may prefer to maintain two distinct categories, one for friends and one for co-workers. Even worse, users change their mind on how they use *vague* categories such as “friends”, resulting in inconsistencies even within a knowledge repository created by a single user; for example, Alia may decide, down the road, that her list of friends (which includes initially co-workers) has grown too big and that she wants to redefine the meaning of “friends” to exclude co-workers.

## 2 SEMIOTIC MODELS

Semiotic fluctuations due to the erratic semiotic behavior of users, illustrated by the example above, should be tracked instead of being smoothed-out. A system that represents personal knowledge should adapt to the idiosyncrasies of each user rather than imposing an average, universal, same-for-everybody interpretation of vague terms such as “friends”.

### 2.1 Signification Events

This motivates the adoption of a more refined knowledge model, which we call a *semiotic knowledge model*, whose primitives are the *signification events* which occur when a *symbol* is paired up with a particular interpretation or *instantaneous meaning*. Knowledge is intimately related to representations: known things, facts, events, situations, rules and laws are those for which an *agent* possesses an internal representation. Internal representations rest still in some repository, providing *static* knowledge, until they are recruited by a signification event yielding a fragment of *dynamic* knowledge, which is the manifestation of the representational activity of the agent.

A signification event (SE) is somewhat related to what semioticians call a *sign*, which comprises something, called a *signifier* (or *symbol*) which stands for something else (*signified*), the represented entity (Chandler, 2007).

*Speaking* results in signification events. Consider the following example. John is at his desk chatting with his friend Mary over the internet. Suddenly, a mouse jumps on John’s desk and John tells Mary: “There is a mouse on my desk!”. Mary replies: “What’s new, there is always a mouse on your desk!”. The word “mouse”, a symbol, yields (at least) four SEs in this exchange; a first one, which is an *efferent* SE, occurs when John maps an internal mental representation of the rodent he has just seen to the word “mouse”; a second SE, which is an *afferent* SE, occurs when Mary hears “mouse” and maps this word to an internal mental representation of a computer device; a third one (efferent) occurs when Mary utters “mouse” and a fourth one (afferent) occurs when John hears “mouse”. The instantaneous meaning in the first SE is a rodent, whereas it is a computer device in the last three (assuming John understood the intended meaning of Mary’s sentence).

Note that the utterance of a sentence involves a *burst* of signification events corresponding to the grammatical components of the sentence: “desk”, “my desk”, “on my desk”, “a mouse on my desk”, and the whole sentence “There is a mouse on my desk” all yield signification events.

To better visualize a semiotic model of knowledge, it may be useful to assign space-time coordinates to SEs which identify the location of the *agent* at which the SE occurs (and perhaps even the specific location within an agent where the representation of a symbol is stored) and the time at which the SE occurs. A SE becomes then a *semiotic point*, where the term *point* indicates, in addition to its space-time embedding, its primitive and atomic nature as a constituent of signification and dynamic knowledge: the instantaneous co-presence of a signifier and a signified is the minimum requirement to establish a representation and a fragment of knowledge. The ensemble of semiotic points yields the *semiotic field*.

The semiotic knowledge model described here represents semiotic points by immutable symbols called *semiotic point representations* (SPR) and uses these to build dynamic and adaptive *semiotic structures*, which can represent all types of information elements (categories, named entities, lists, properties, relationships, facts, facts about facts, etc.) by adopting and tracking over time each user’s terminology and organizational schemes.

In the context of personal knowledge representations where several people (e.g., the members of a family) share the same database and contribute information to it, a basic SPR can be constructed by concatenating (1) an identifier for the

symbol involved in the SE; (2) an identifier of the information contributor; (3) a timestamp of when the information was entered. This basic SPR can be enriched by any kind of available contextual information to yield a more informative SPR (e.g.: the place where the contributor was when he/she entered the information, his/her mood, etc.).

## 2.2 Modes of Operation

Users enter information into the system in one of several ways: (1) *Direct* entry method: users speak or type information such as primitive linguistic symbols (words, names, etc), primal symbols (numbers, dates, physical quantities) and simple natural language expressions; (2) *Compositional* entry method: users select existing symbol by browsing the repository and compose them into new composite symbols, for example, by manipulating widgets displayed on a screen via a drag-and-drop interface. The compositional method, which produces *structured linguistic expressions* (SLEs), described later, minimizes and simplifies the recognition step since the user utilized symbols already known to the system. (3) *Analytical* method: *Unstructured* linguistic expressions, such as “It rains today”, entered directly into the system, must be converted into SLEs, either manually, or with the assistance of a suitable natural language processing module.

By combining the above entry methods, users can introduce their own terminology into the system and reuse it. In addition to these entry methods, users can operate at the level of semiotic structures to update, revise, organize and re-organize their information. Users can also import (portions of) external dictionaries and ontologies and adapt them to their personal ones.

## 2.3 Ingestion, Matching, Inference

(Balog & Kenter, 2019) point out that entity linking, population of the repository and detection of new entities (nil-detection) are intertwined in personal knowledge representations: the semiotic model proposed here comprises a symbol matching component which carries out these three steps in a unified fashion every time the repository system *ingests* a packet of information delivered by the user. The repository acts like an active memory and attempts to recognize every symbol occurring in the burst of signification events (SEs) produced by the input data; every detected input symbol is matched against the symbols stored in memory. Different methods and data structures are used for symbol

matching: *K-nearest neighbours* algorithms to search primal symbols embedded in metric spaces; “vertical” *compositional hypotheses lists*, obtained by tracking and recording the usage of symbols in composite symbols, to detect potential matches; “horizontal” *relaxation*, to deal with noise by extending the search to neighbouring symbols; and rule-based *reasoning*, which attempts to infer searched symbols from existing symbols.

It is well known that there is a trade-off between the expressive power of a representation language and the mathematical computational properties of its reasoning capabilities (Suchanek, 2020). The representation language adopted here, defined by *parametric symbols* and the structured linguistic expressions (SLEs) described later, does not place any restrictions on the kind of information that users can try to communicate to the knowledge repository. Reasoning is viewed here more as a sequence of *internal signification events* than a crisp logical computation; its grounding is more of a statistical nature than an axiomatic one, so that issues of plausibility and defeasibility of the derived results take precedence over decidability. As in many knowledge models based for example on modal logic or fuzzy logic, *truth* is not an absolute value, and the goal of reasoning is more one of growing large *regions of coherence* (both *logical* and *semiotic* coherence) than one of deducing true facts missing from the repository. An *inference graph* method, which complements the search method by using rules to complete partial matches detected by the search module, will be described elsewhere.

## 2.4 The Symbol Abstraction Hierarchy

The term “symbol” is used here in a quite broad sense and includes both “concrete” and “abstract” symbols. Three abstraction levels for symbols are considered: *materialized* (or *concrete*) symbols, and two levels of abstract symbols: symbol *forms* and *multiform symbols*.

**Materialized Symbols** are physical embodiments of symbols and are physically connected to the signification events (SEs) which they trigger, or by which they are created or “activated”. Materialized symbols occupy a region in space-time. For example, a road sign positioned near a road intersection triggers signification events whenever someone sees it; a neuron or a pool of neurons in the visual cortex of a primate is a materialized symbol which is activated when a particular visual stimulus is presented; a chunk of memory cells in a computer holding a digital

representation of a word triggers signification events whenever a computer program reads the memory cells and acts on it based on a meaning assigned to the word. *Distributed* representations also have materialized embodiments; however, they are not uniquely identified by space-time coordinates since multiple distributed representations share the same region in space-time. *Mental* representations (which are arguably distributed representations) also have materialized embodiments, even though it is not always clear what they are.

**Symbol Forms.** It is convenient to (conceptually) group a collection of materialized symbols which are invariant under some transformation into a *symbol form*. Symbol forms are abstract symbols, where “abstract” means here non-physical, that is, not localized in space-time. Examples of symbol forms: (1) the equivalence class of all instances of the character string “mouse”, across all printed documents. (2) The equivalence class of all stored bit sequences representing the string “mouse” in a computer (3) The equivalence class of all visual representations of the string “mouse” on a computer screen.

**Multiform Symbols.** As illustrated by the previous example, what is commonly called a symbol, such as the word “mouse”, is typically associated with multiple forms: printed form, digital form, displayed form, etc. Hence it is called a *multiform symbol*.

Multiform symbols are a useful concept to build efficient software implementations of knowledge representations. Specifically, with an object-oriented programming language (OOPL) such as Java, it is possible to construct objects which provide detailed representations of any kind of entity. These objects can be quite complex, and their size can be very large; think for example of the list of all filenames of someone’s digital photos. It makes sense then to introduce more compact representations of these objects.

The most compact representation *form* of a symbol is arguably an integer which identifies the memory address of a record which defines the symbol (this is roughly how Java represents the values held by variables). *Composite* symbols built from *constituents* can be represented by the array of integer identifiers of its constituents, called the *compositional code* of the multiform symbol. For example, a large list of file names can be represented more compactly by representing each file name with an integer rather than a string containing the file name. The full-

fledged form of the object, given by an object wherein all nested identifiers have been expanded, is called the *exploitable* form of the symbol. *Hybrid* compositional codes contain some constituents in identifier form and some in exploitable form. Optimized *variable-complexity representations* are obtained by cleverly managing the computational forms of a symbol so that only those components that are needed in exploitable form are expanded.

The different forms of a multiform symbols all have same *putative* (assigned) meaning although they do not have the same *intelligibility* and *exploitability*: *form conversions*, which are semantically invariant, are needed to enable efficient computation and communication.

## 2.5 Primal Symbols

Semioticians distinguish between three types of symbols: indexical, iconic and conventional (Chandler, 2007). Indexical ones are those which are physically or causally linked to their signifiers, whereas for conventional symbols the link is established via conventions (iconic symbols are not discussed here). An analogous distinction can be made between *primal* and *linguistic* symbols. Whereas linguistic symbols are plagued with all the issues due to the conventional nature of their meaning (ambiguities, redundancies, context-dependence of the meaning, vagueness), *primal* symbols are those which are assumed to be free from these problems. Every symbol of a mathematical nature (numbers, real vectors, etc.) or issued from a formal language is a primal symbol. Physical quantities, such as 3kg, and standard-defined entities, such as GPS coordinates, are primal symbols. Character strings, stripped of any linguistic meaning, can also be treated as primal symbols.

Primal symbols usually belong to metric spaces whose metric structure is important for search and matching. For example, K-nearest-neighbours algorithms can be used when searching for a match to a primal symbol, and ad-hoc clustering algorithms can be used for organizing and grouping occurrences of primal symbols. It seems appropriate to store and maintain primal symbols in ad-hoc memory *slices* or databases where these operations can be carried out more efficiently.

## 2.6 Structured Linguistic Expressions

Natural languages are arguably the most powerful representation systems and can be used to represent all types of entities. The term “entity” is used here in



the broadest sense possible: anything which can be referred by a linguistic expression is an entity, whether it “exists” or not. Things, stuff, living beings, abstract concepts, relationships, statements, facts, facts about facts, rules, etc., are entities which can be represented (signified) by a symbol, notably by a linguistic expression, which is encoded here as a *Structured Linguistic Expression* (SLE).

One simple way to obtain an SLE is to begin with an unstructured linguistic expression, that is, a character string containing a natural language expression, and to build a *parametric linguistic symbol* representing an n-ary *predicate* or *relation*, by replacing one or more fragments of this character string with “fillable slots”. For example, from: “There is a mouse on my desk”, one can build the parametric symbol “There is #1 on #2”, obtained by replacing “a mouse” and “my desk” with fillable slots denoted #1 and #2.

A parametric symbol can be viewed as a lambda expression or a function which maps tuples of symbols to a symbol. We use the notation  $\langle * \rangle$  to indicate such a function, and  $| * \rangle$  to denote the arguments passed to the function. We further associate an integer identifier to these symbols so that the construction of an SLE encoding for our example can be represented by the following “script”:

```
$0< There is a mouse on my desk>
$1< There is #1 on #2|
$2|a mouse>
$3|my desk>
$4[1,2,3]< There is #1 on #2|a mouse, my desk>
```

The multiform symbol with identifier \$4 has a compositional code form [1,2,3] and an exploitable form  $\langle \text{There is } \#1 \text{ on } \#2 | \text{a mouse, my desk} \rangle$  as indicated by the last line of the script.

The symbol \$4 is an SLE *encoding* of the original unstructured expression \$0; note that even though they can be declared to be semantically equivalent, \$0 and \$4 are two distinct multiform symbols.

Unary predicates, such as  $\langle \#1 \text{ is a mouse} \rangle$ , can be converted into *category* symbols denoted, for example,  $\{ \#1 \text{ is a mouse} \}$ . An equivalent symbol for this category is  $\{ \text{mice} \}$ . Conversely, a category can be converted to a predicate, for example  $\langle \text{mice} \rangle$  is a predicate equivalent to  $\langle \#1 \text{ is a mouse} \rangle$ .

*Samples* of categories are obtained by specifying N arguments which are typically the names associated with the items in the sample; for example, two mice named “Jerry” and “Billy” can be represented as  $\{ \#1 \text{ is a mouse} | \text{Jerry, Billy} \}$  or  $\{ \text{mice} | \text{Jerry, Billy} \}$ . A category sample can also be built by specifying an integer which indicates the number of (unnamed) items in the sample; for example, “a mouse” is

encoded as  $\{ \text{mice} | 1 \}$ . Category sample can also be used to denote quantities of “stuff”, for example  $\{ \text{water} | 1 \text{ litre} \}$  denotes one litre of water.

Categories can be used to restrict the allowed slot fillers of a parametric symbols, for example:

```
<The capital of <states|#1> is <cities|#2> |
is a restricted parametric symbol.
```

Note that SLEs can contain other SLEs as constituents, which gives users practically unlimited expressive power; facts about facts, beliefs, etc. can be easily expressed as an SLE, for example, “João believes that it will rain today” can be encoded as:

```
$0<it will rain today>
$1<#1 believes that #2 | João, $0>.
```

On the other hand, unless constraints are introduced, it is possible to build nonsensical SLEs and statements such as  $\$0 \langle \$0 \text{ is false} \rangle$ , which cannot be assigned a truth value.

## 2.7 Symbol Groupings

In addition to linguistic symbol composition (obtained via parametric symbols) and category samples (normally obtained via lists of items), symbols are grouped according to the following organizing principles. (1) *Topological* groupings are obtained by grouping symbols related by proximity or similarity. These groupings provide *relaxation regions* used when searching for symbol matches; for example, when searching for a word, a synonym found in a relaxation region can be returned as a valid potential match. *Clusters* in primal metric spaces are also topological groupings. (2) *Inferential* groupings are obtained by grouping symbols which participate in an inferential derivation (e.g., a syllogism). These groupings provide a *justification* for an inferred symbol and can be useful to assign plausibility scores and to compare alternative or incompatible states of affairs. (3) *Descriptive* aggregates enhance the description of some entity by combining, for example, SLEs which refer to a common entity. For example,  $\$1 \langle \$0 \langle \text{Li} \rangle \text{ lives in Boston} \rangle$  and  $\$2 \langle \$0 \langle \text{Li} \rangle \text{ likes movies} \rangle$  can be grouped into a descriptive aggregate  $\{ \$1, \$2 \}$ . As another example, a category symbol can be aggregated with restricted parametric symbols which use that category as a *restricting category*, so that a user can be presented with a list of available statement builders applicable to members of that category. Note that from the point of view of knowledge graphs, a descriptive aggregate can be viewed as the *reification* of a star-shaped subgraph centered at the “described” entity.

**Taming Complexity.** Symbol groupings play an important role for (1) building a flexible and incremental representation system; (2) representing the “topology” of symbols (which, in turn, is essential to make the system *robust*) and (3) making search and matching with relaxed compositional hypotheses lists more efficient by gathering an entity’s features under a unifying symbol identifier. However, they also introduce a great deal of complexity if they are allowed to proliferate in an uncontrolled way. One answer to this problem is to organize the multiple manifestations of an entity (including symbol groupings), which emerge from the interactions with users and their data, into *semiotic structures*, as described later in more detail.

A second difficulty arises from the potentially very large size of the computational objects needed to represent complex symbolic constructs, such as symbol groupings. Think for example to a computer program that maintains a table to represent all the files or all the emails of a user: it is clearly impossible to hold such objects permanently in live memory. Multiform symbols and variable-complexity representations play a role here by converting each symbol to the most appropriate computationally optimized symbol form, ranging from simple integer identifiers to full-fledged exploitable forms (with hybrid compositional code in between).

## 2.8 Contextualized Symbols

In the analysis of documents, a distinction is made between *word types* and *word tokens*. We generalize this distinction by considering word types a subclass of *de-contextualized* or *context-free* symbols and word tokens a subclass of *contextualized* symbols. Words such as <mouse> and sentences such as <There is a mouse on my desk> are context-free symbols because no context is specified.

A contextualized symbol is one for which some contextual information is specified.

*Plucked* symbols are one type of contextualized symbols. They are obtained by “plucking” a constituent symbol from a composite symbol. For example, the occurrence of \$0<a mouse> in

\$1<#1 ran across my desk | \$0<a mouse>>, denoted by the *symbol coordinate* \$1.1 (\$n.k denotes the k-th constituent of \$n) is a contextualized symbol because a context has been specified for the symbol \$0, namely a sentence in which it occurs. Note that \$0 and \$1.1 are quite different symbols: \$0 can be either rodent or a device, whereas \$1.1 is (most likely) a rodent.

**Semiotic Point Representations (SPR).** Semiotic point representations, which represent signification events, are contextualized symbols. Recall that a signification event occurs when a materialized symbol gets connected to a referent or signified. The sequence of signification events triggered by a materialized symbol corresponds to a sequence of time samples of the spatial region occupied by it. For example, a road sign planted at a crossroad is “sampled” every time someone sees it and understands its meaning (and also when the sign is misinterpreted).

There are different types of SPR which differ in the amount of information they convey about the signification event. A *bare* semiotic point identifier simply identifies a signification event by providing the space-time coordinates that uniquely identify it but does not convey any information about the meaning conveyed. For example, specifying the GPS coordinates of a road sign and the times at which it has been seen identifies a sequence of signification events, but does not provide any information about what the sign meant to those who viewed it.

An *informative* SPR is one which does provide useful information to recover the instantaneous meaning of the represented signification event. A typical informative SPR specifies the symbol involved in the signification event plus some contextual information that restricts its possible interpretations. In the context of information extraction, *word mentions*, which are often associated with a few surrounding words, can be viewed as informative SPRs. A word along with the sentence in which it appears does not, however, specify a unique semiotic point since the word and the enclosing sentence are created once, yielding an initial semiotic point; and then read multiple times by different readers, yielding many additional semiotic points.

## 2.9 Semiotic Structures

All multiform symbols discussed in detail up to now, that is, primal symbols, parameterized symbols, SLEs (including categories and category sample), symbol groupings of various kinds, SPRs, are *immutable* symbols. Once created, they can be stored permanently in a repository and assigned an identifier which can be used confidently to refer to the symbol, with a guarantee that the symbol does not change, except for semantically invariant form conversions. For this reason, they can be called *stock* symbols.

It should be noted that an immutable symbol is not one whose meaning is necessarily immutable (unless it is a primal symbol). For example, the symbols

<today> and <you> are immutable but their meaning clearly depends on the context in which they are used.

*Semiotic structures* are mutable symbols. A semiotic structure designates a *representative symbol* which can be replaced by another representative symbol when the semiotic structure needs to be updated. An identifier of a semiotic structure identifies a symbol which can change over time, differently from identifiers of immutable symbols. There is a crisp and fundamental distinction between immutable and mutable symbols.

We now describe some of the ways a semiotic structure evolves over time, usually with the active intervention of a user.

**State Updates.** The list of my friends must be updated every time I meet a new friend. To enact this update, a new immutable symbol representing the new list of friends is created and the semiotic structure representing “my friends” is updated with the new list, which becomes the new representative symbol of the semiotic structure (the new list can of course be represented more efficiently as an edit to the old list if, for example, only one friend has been added). This type of update is called a *state update* because it reflects a state change in the underlying entity represented by the semiotic structure. As another example, a semiotic structure representing the GPS coordinates of my car must undergo a state update every time the car moves.

**Informative Updates.** State updates should be contrasted with *informative* updates, which occur when new information is provided about the represented entity (and the underlying entity does not change). For example, suppose a user Xiu, while tagging pictures with people’s names, has quickly introduced a new person into the database by a simple keyword such as \$0<Chen>. Later, Xiu realizes that she knows two Chen’s, so that she needs to provide a more informative representation of the first Chen, for example, by means of the more informative symbol \$1<coworkers|Chen>. This informative update is also a **refinement update** because not only does it provide additional information about the entity (that is, that Chen is a co-worker), but it also reduces the ambiguity of the symbol, hence *refining* the set of objects it may refer to.

A purely informative update which does not reduce ambiguity is referred to as a **descriptive update**. For example, suppose Maria knows only one Pablo but she wishes to enhance her representation of Pablo by including his phone number. In a knowledge

graph this would be done by adding one edge; in the currently proposed model this could be done creating an SLE such as \$2<The phone number of #1 is #2 | \$0< Pablo >, (617)-123-4567> and then by plucking “Pablo” from this SLE, to yield the plucked symbol \$2.1. An alternative way is to create a descriptive aggregate which *annotates* \$0 with \$2.

**Renaming updates** are used for renaming an entity; for example, if Maria decides to rename Pablo to “Pablito”, the current representative of the semiotic structure, say <Pablo>, is replaced with <Pablito>.

**Consolidation Updates.** Suppose that after using the repository for a while, Maria has mentioned Pablo multiple times so that the repository now contains multiple SLEs having <Pablo> as a constituent. While browsing through the repository and seeing multiple mentions of Pablo, Maria decides to gather all the information about Pablo in one place and creates a descriptive aggregate about Pablo: this new descriptive aggregate becomes the new representative of the semiotic structure representing Pablo. Note that a consolidation update is an informative update. Consolidation updates play a crucial role in taming the proliferation of symbols referring to the same entity and ought to be triggered automatically by the system when necessary.

**Splitting.** As shown earlier, fluctuations arising with the use of vague symbols such as <friends> may lead to inconsistencies. *Nested* semiotic structures can be used to track these semiotic fluctuations and to represent the structure of vague entities. Specifically, an initial semiotic structure representing a vague symbol such as <friends> can spawn two nested semiotic structures by creating two refinements of <friends>, one which includes co-workers and one which exclude co-workers. These two refined symbols become the initial representatives of the two nested semiotic structures, each of which inherits one portion of the semiotic history of the original vague symbol.

**Merging.** Two semiotic structures whose meanings are similar or overlapping can be combined into a coarse semiotic structure. For example, suppose that a group of family members has independently kept lists of <people> but they have (unconsciously) assigned slightly different meanings to the symbol <people>. For example, Alia has excluded fictional character, such as Harry Potter, from her list (she has a separate category for them); Elif has also excluded people he has never met (he has a special category

“celebrities” for famous people he has never met); Canan has instead included all speaking entities into his list including cartoon characters such as Tom the cat and Jerry the mouse, just because he was too lazy to create a special category for them. It is finally decided to consolidate the databases of all family members and for doing so, a coarse semiotic structure named <people> is created and the 3 <people> categories appear as nested categories within the coarse <people> category. The flavors of <people> “invented” by each family member is not lost and is now organized under a vague category symbol that acknowledges each one of three interpretations.

**Extension-by-Reference and Rewinds.** A semiotic structure not only keeps track of the updates of its representative symbol but also of the references to its current representative symbol from other symbols. In other words, a semiotic structure maintains an historical record of all the signification events in which it participates. A user may use this history to fine-tune the meaning of a symbol. For example, the meaning of a symbol may *drift* over time and a user may realize that the a past usage of the symbol represents better its current intended meaning than the last occurrence of the symbol: a **rewind** operation is then executed.

## 2.10 Reflection

Semiotic structures are *meta-symbols*: the history of signification events they contain (in the form of SPRs) represent symbols representing something. The elementary operations just described, occurring at the semiotic structure level: state updates, information updates, refinements, splits, merges, extension-by-reference, rewinds, consolidations, etc., accompany every population step and curation step.

To carry out any of these operations, an agent (human or machine) must engage in *reflection*, which involves recollecting or reconstructing the past meanings of symbols. Reflection results in a *semiotic link* being created between a current signification event and a past signification event which have been assessed by an agent to pertain to the same underlying entity. The two linked signification events become then part of the same semiotic structure.

A *linear* semiotic structure is obtained when an agent always determines that the current meaning of a symbol is the same as its previous occurrence. Bifurcations occur when rewinds are necessary due, for example, to semiotic drift.

## 3 CONCLUSIONS

A knowledge representation model has been described which enables users to enter, modify organize, re-organize, and curate their own personal information by leveraging their own terminologies and organizational schemes. We believe that ideas discussed in this position paper can be used to develop an interactive repository with a user-friendly interface to store and recall personal information. One should also explore the possibility of using semiotic structures (and other symbolic representations described here) to enhance knowledge graphs in general, for example, to represent knowledge extracted from large amounts of text.

## REFERENCES

- Balog, K., & Kenter, T. (2019). Personal Knowledge Graphs: A Research Agenda. *ICTIR 2019*, (pp. 217-200). Santa Clara, CA, USA.
- Balog, K., & Kenter, T. (2019). Personal Knowledge Graphs: A Research Agenda. *ICTIR 2019*, (pp. 217-200). Santa Clara, CA, USA.
- Balog, K., Mirza, P., & Skjaeveland, M. G. (2022, June). Report on the Workshop on Personal Knowledge Graphs (PKG 2021) at AKBC 2021. *ACM SIGIR Forum, Vol. 56, No. 1*.
- Chandler, D. (2007). *Semiotics*, Routledge. London and New York, 2<sup>nd</sup> edition.
- Ilyas, I. F., & Chu, X. (2019). *Data Cleaning*. ACM.
- Kejriwal, M., Knoblock, C. A., & Szekely, P. (2021). *Knowledge Graphs*. Cambridge, Massachusetts. London, England: The MIT Press.
- Montoya, D., Pellissier Tanon, T., Abiteboul, S., Senellart, P., & Suchanek, F. M. (April 2018). A knowledge Base for Personal Information Management. *LDOW 2018*. Lyon, France.
- Rusell, S., Norvig, P., & Davis, E. (2010). *Artificial Intelligence: a Modern Approach. 3rd eds*. Saddle River, NJ: Prentice Hall.
- Suchanek, F. (2020). The need to move beyond Tripls. *Proceedings of the Text2Story'20 Workshop*. Lisbon, Portugal: R. Campos, A. Jorge, S. Bathia (eds.).
- Weikum, G., Dong, X., Razniewski, S., & Suchanek, F. (2021, August 23). Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases. *Foundations and Trends in Databases*.