AGI engineering

Subscribe

SEMANTIC STORAGE

THE SIMPLER, THE MORE VERSATILE

Mykola Rabchevskiy
Apr 9 2 4

As discussed earlier in chapter #3 on architecture, Semantic Storage keeps relationships between logical entities.

The simplest type of relationship is that an entity belongs to a particular set that unites elements that are similar in some way, that is, belonging to a specific logical category/class. Since such a set in itself is also a logical entity (in what follows, we will omit "***logical***", that is, when we say "***entity***", we mean a ***logical entity***), then this relation under consideration is a relation between an ***entity-element*** and an ***entity-set***, for example:
red ∈ ***color***
ant ∈ ***insect***
mammal ∈ ***animal***
desktop ∈ ***computer***

In natural language, this relation expressed by the phrase "***A is a B***":
red ***is*** *a color*
desktop ***is*** *a computer*

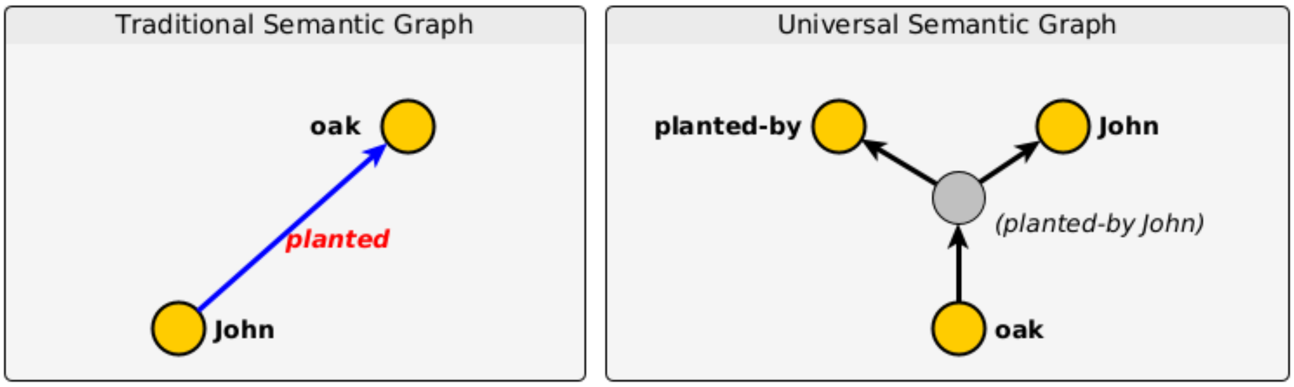
Mathematics to express the same relation uses the operator of the element's membership in the set:

of an ***element of a set***, and in another case, play the role of a ***set of entities***. Two roles of the same entity "***mammal***":
mammal : animal
cat : mammal

In what follows, an entity that can be used as a set will be called a ***concept***, and an entity that cannot represent a set of entities will be called an ***atom***. Those familiar with a data structure called "***tree***" will find an analogy with the tree's ***nodes*** and ***leaves***.

Graphically, the relationship "***is***" can also be represented as a ***Venn diagram*** or as an ***edge*** of a ***directed graph***. The vertices of the "classical" ***semantic graph*** are associated with entities, and the edges are associated with ***relations***. This approach, however, is good as long as we operate with a fixed set of relationships. In a system that assumes the versatility level required from an AGI system, the requirement to use a predetermined unchanging set of relations is obviously unacceptable.

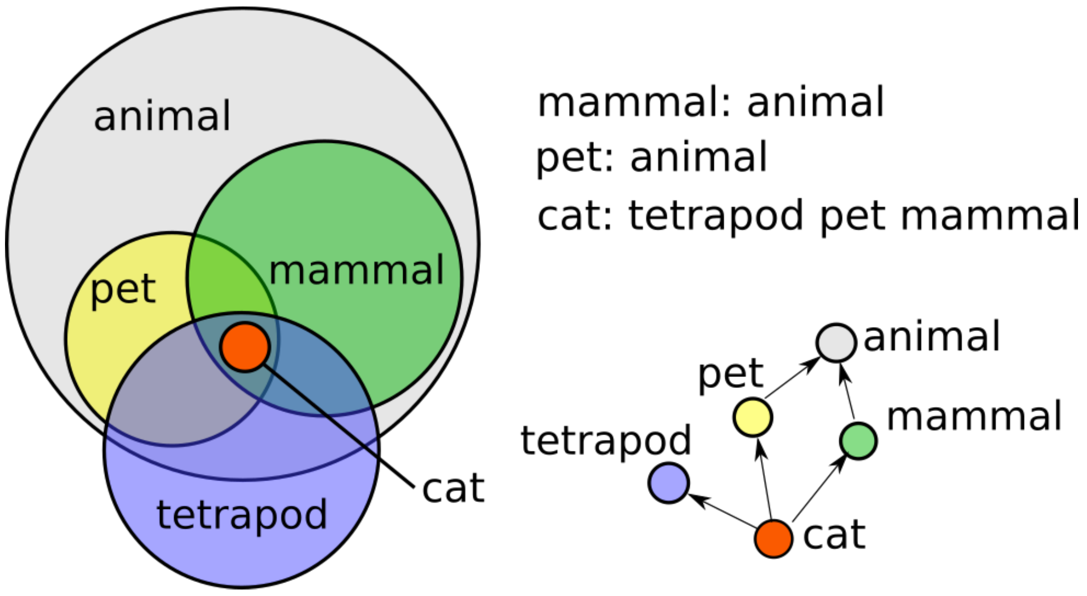
But the expandable set of relations means that the relations become part of the set of concepts that the AGI system operates with. In the semantic graph, both vertices and edges begin to be associated with entities. This mentally confuses the situation in terms of logic and complicates the software implementation of semantic storage. This can be avoided by the possibility of ***reducing any relations to a single relation `is`***. Examples of such a reduction:
John planted the oak <-> *The oak* ***is*** *planted by John*
Ann wants to buy a house <-> *Ann* ***is*** *a house owner wannabe*



It is easy to see that a combination of words represents the right side in both cases. Nevertheless, each of these combinations is essentially a ***reference to a concept***. Since the concepts on the right side are interpreted as sets, their combination can also be interpreted as a set, similar to how in algebra, the combination "***a b c***" is interpreted as the product of the quantities ***a***, ***b***, and ***c*** (but in our case the sequence of concepts determines the ***intersection of the sets***). Accordingly, the concepts' sequence represents an ***unnamed concept***, and the chain of concepts "***house owner wannabe***" is an ***expression*** in our ***concepts manipulating language***.

If all relations are reduced to a single relation "***is***", all edges of such a graph must be labeled in the same way according to the rules adopted for semantic graphs. Obviously, this simply turns the “classical” semantic graph into a ***directed graph with unlabeled edges***. Since such a graph allows one to describe and classify entities of all sorts, we will call it a ***universal semantic graph*** (to distinguish it from the traditional semantic graph).

The graph's edges are directed from the entity on the left side of "***is***" to the concept on the right side. If the right-hand side is given by several concepts that determine the intersection of the corresponding sets, then a fragment of the semantic graph contains edges going to each of the right sides' vertices. It is often possible to specify an entity without naming it explicitly, but using the fact that the sought entity is the only element in the intersection of several sets, each of which is specified by a named concept.



We will follow this rule: if ***A***, ***B***, and ***C*** are some concepts, then the chain ***A B C*** defines a set of entities ***e***, for each of which the statement "***e is A***, ***e is B. and e is C***" is true. And putting the chain in brackets: "***(A B C)***," we get a ***reference to the entity*** if the expression inside the brackets defines a set of a single element. Thus, the parentheses change the interpretation of the expression using dualism, as mentioned above.

The use of anonymous (nameless) entities in the AGI system increases the implementation efficiency (there is no need to name each entity). It also makes expressions in the concept manipulation language more similar to phrases in a natural language, which de facto uses this principle.

If **e** ∈ **S**, that is, if "***e is S***" is true, we will call the concept ***S*** an ***e's sign***, and the ***complete set of signs of the particular entity*** will be called an ***entity syndrome***. For example, in the statement
cat : nice mammal pet
"***nice mammal pet***" is the syndrome of the entity "***cat***" consisting of three signs "***nice***", "***mammal***", and "***pet***" (if there are no other signs assigned). Since the syndrome is a set of signs, changing their listing order does not change the meaning. In terms of the graph, the entity's ***syndrome is a set of outgoing edges*** from an entity vertex. Similarly, the ***set of ingoing edges*** defines an entity's ***explication***, that is, the set of entities covered by the concept. The functions of obtaining the syndrome and the explication of the entity ***e*** will be denoted by ***S(e)*** and ***E(e)***, respectively.

Thus, we can describe a fragment of a semantic graph with a phrase (statement) in the language of concept manipulation, and the ***graph as a whole is equivalent to a collection of such statements***. This equivalence is the foundation of the textual interface for the Semantic Storage.

The role of the single primary relationship "***is***" is reminiscent of the role that binary numbers play in the computer world, to which all other kinds of data are reduced.

The effectiveness of the software implementation of semantic storage significantly depends on the selected data structure. As practice shows, both speed of execution and amount of required memory improved when the graph is represented by a hash table, in which the IDs of the vertices (entities) are used as keys. The corresponding data are entities` syndromes, that is, the sets of nodes to which the outgoing edges from the key-vertex are directed. In this case, the set of elements of the syndrome are, in turn, represented by hash sets. That is, in terms of C ++ STL (Standard Template Libray), an object of the type can represent a graph

unordered_map< ***Identity***, ***unordered_set***< ***Identity*** > >

where ***Identity*** is an entity's ID type.

Implementation efficiency is also increased by using random variables as entity IDs instead of computed hash keys and using hash table variants that allocate memory for the entire table once (flat hash tables).

There is a reason to store names of entities separately from the graph itself - firstly, because most of the vertices in real applications remain unnamed (anonymous entities), and secondly, because the same graph can be associated with different languages.

Storing the ties of vertices with names as a separate bi-directional glossary (ID -> name and name -> string) allows you to switch the interface from one language to another and use several languages simultaneously. Naturally, bidirectional dictionaries are effectively implemented by hash tables.

In addition to the semantic graph itself and these glossaries, it is helpful to have one more subcomponent, which stores ***sequences of entities*** associated with the graph's vertices. Suppose an entity is essentially a ***sequence of actions*** (routine), and actions are entities represented in a graph. In that case, it is practical to use a hash table that maps a concept to a corresponding sequence of entities. Another type of sequence is ***temporal patterns***. Regardless of meaning (routine, temporal pattern,..), all sequences stored in the single hash table - are united not by the substance but by the fact that the elements are ***ordered*** (unlike the sets ***unordered by their nature***).

An important aspect is the ability to manipulate semantic information in terms of sets - a set of entities corresponding to a particular concept (a group of colors corresponding to the concept of "color") and entity syndromes as a set of concept signs. At the same time, the underlying data structure is a directed graph with unlabeled edges.

A set of operations (functions) for manipulating semantic information is the subject of the next chapter.

SUMMATION

- *Each vertex of the semantic graph is a logical entity.*
- *Duality is innate in logical entities when an entity can be used both to denote a set and as an element of a set.*
- *All relations are reduced to the base relation "**is**".*
- *The semantic graph's edge is directed from entity to concept-sign and is not associated with anything (unlabeled).*
- *Entities and corresponding graph nodes may not have a name (anonymous entities).*
- *The number of entities represented in the graph is usually much larger than the number of named entities.*
- *A semantic graph is essentially a collection of statements.*
- *The language of concept manipulation is a variant of natural language formalization.*

- *The grammar of the semantic information manipulation language does not contain any reserved words.*

← Previous Next →

LeBoeuf Wellington May 9 Liked by Mykola Rabchevskiy

The placement of cat in the Venn diagram suggests that all cats are pets. I think the placement needs to move to a position over the pet boundary. This will indicate that some cats are pets and some are not. Further, I imagine that the Venn diagram suggests that not all relations reduce to 'is' because 'is' infers an affirmative positive semantic; whereas we also need affirmative negative semantic ('is not') to complete our understanding; noting that the absence of 'is' is insufficient to infer 'is not' and will only produce an unknown.

1 Reply

1 reply by Mykola Rabchevskiy

supralibrix@gmail.com Apr 17 Liked by Mykola Rabchevskiy

This is parallel to my work.

Let's connect.

https://physix.world/

1 Reply

2 more comments…

Subscribe