A Minimalist Framework for Ontologies

Julius Hamilton

November 26, 2024

1 Motivation

The idea behind this format for personal knowledge bases is that humans have "thoughts", and thoughts can be *intentional*, or, "about other thoughts". Call this "aboutness", if you like.[1] The way that one thought can be "about" another thought will be captured by a function which associates one thought to another, called associate. An element of the knowledge base will be of type Element:

$$\mathsf{associate} : \mathsf{Element} \times \mathsf{Element} \to \mathsf{Element}$$

The association function generates a free magma. A subset of this magma is a knowledge base.

For example:

generators
"dog"
"pickle"
"I like music"

elements
associate("dog", "I like music")
associate("I like music", "dog")
associate("pickle", associate("I like music", "I like music"))
...

knowledge base
associate("this makes no sense", associate("dog", "pickle"))
associate("I have a dog", "dog")
associate("I like jazz music", "I like music"))
...

[1] https://plato.stanford.edu/entries/intentionality/

2 Operations

One of the first most useful tools we have at out disposal is asserting *equality* between two terms, in this "algebra". For example, if for my intents and purposes, I do not wish to distinguish between the capitalized and uncapitalized version of a friend's name, I can add an equation to the 'theory':

"Gabe Ross" = "gabe ross"

Equality will be closed under reflexivity, symmetry, transitivity, and subterm replacement, so that the following formula follows from above:

associate("he's a friend of mine", "Gabe Ross")

=

associate("he's a friend of mine", "gabe ross")

From here, I want to begin showing how you can run various kinds of queries on this. One of the points of this data model is that it's "infinitely relational": you can always add in a new relation, to relate two terms. In general, since every term is a pair, there should also be useful ways to manipulate the data with "key-value" type queries.

I also want to show how relations can be "internalized". For example, instead of an equality symbol, what if instead, I used an internal relation, "that's the same person", like so:

associate("that's the same person", associate("gabe ross", "Gabe Ross"))

One of the first operations I will explore defining could be called "merge". It would act something like this. If I have elements:

"dog", "horse", "hyena", "dolphin"

Then I would like somehow to define "classes of things", maybe by allowing sets of elements, something like:

associate("these are animals", {"hyena", "dog", "dolphin", "horse"})

We can imagine operations which allow us to take the union of elements with the same key, for example,

union(

associate("this is an animal", "dog"),

associate("this is an animal", "horse"),

associate("this is an animal", "hyena"))

$\rightarrow$

associate("these are animals", {"hyena", "dog", "horse"})

(and also the reverse: "splitting" such elements.)

Ultimately, the "keys" should allow us to make queries, like, "Is "dog" an animal?", just as a simple example for now.