On ontology

15.10.2017 Arve Meisingset

This note is inspired by a presentation of DOLCE, Descriptive Ontology for Linguistic and Cognitive Engineering, by Laura Slaughter. This note does not convey the presentation.

Both Set theory and mereology try to define ontological structures that deviate from the syntactical structure of the statements describing these ontologies. This belief, that they are different, is the cause of their problem. They believe that reality and concepts are fundamentally different from data.

I am a phenomenologist. The phenomena exist inside an observer. The phenomena – as well as the external (formal) language – are made up of data, and the mappings between them are data, as well. This changes the entire discussion.

On logics and ontologies

Some ontologies study what exist in the real world, others study concepts.

In phenomenology, a phenomenon exists if it is observed by an observer. My book, section 4, contains a detailed list of requirements for accepting something to exist.

The phenomenon is inside an observer. The phenomenon is data inside this observer. The phenomenon may be contained in a photo in a camera. Hence, we consider the phenomena to be organizations of pixels, independently of what kind of means are used for the observations. The organization of pixels is a list of lists. Each element is one pixel, and each phenomenon is one pixel, which may contain more pixels.

In phenomenology, we consider any entity that is impacted by something else, to be an observer. A Garage is observing a Car who is observing an Owner. The Owner is a phenomenon that requires that a corresponding Person phenomenon exists. This is expressed in Existence logic through a condition combined with navigation.

We make four comparisons to classical logic:

1. In Relational mathematics, and often in Predicate calculus, they believe that the role Owner has the same identifier as the entity Person, and that they have the same attributes. In phenomenology, we consider the Owner and Person to be two different Phenomena, with their own attributes.
2. In Relational mathematics, and often in Predicate calculus, they consider an entity to be identified by a term. In Existence logic, the entity is identified by a complex expression of a path with sub-branches.
3. In Relational mathematics, and often in Predicate calculus, references are made by names. In Existence logic they are made by navigation. References by names are similar to non-locality in quantum physics. References by navigation are similar to local actions, only.
4. In Relational mathematics, and often in Predicate calculus, they use globally unique names only. Similar inscriptions refer to one and the same string, which refers to one set, one entity or one concept only. This is called the Type-token principle; which is so fundamental that it

most often is not axiomatized. In Existence logic, significant duplicates are the normal. Each similar inscription denote a different phenomenon, and we do not deal with strings and sets.

The above lists some of the dis-similarities between classical logic and its ontologies, and phenomenology and its Existence logic. The differences and their impact are essential.

Quote from the Post Script of my book

“In this book, we have learned that observations create threads of phenomena inside an observer. These threads may be combined and played like a movie or dream. See on the Virtual Travels technology in Part 1 section 2. In nature, these threads are expressed as nerve cells, branches of trees, threads of fungus etc.

Descriptions are created by making copies of the threads of phenomena. Both phenomena and their descriptions are made of data inscriptions inside the observer. The data may or may not contain nametags, which themselves may be created by threads. Nametags may be convenient to have, but are not needed. This is very different from traditional formal languages, which rely on unique names of constants, variables, predicates, functions and operators.

Denotations make up a ladder of data between the threads of descriptions and the threads of phenomena. In most cases, these denotation mappings are not needed, but we need to understand how data may or may not reflect phenomena.

In Existence logic, we need only one symbol (:) to state any thread of data and operations on the data. We need no alphabet, proposition, operator or truth-value. Use of Existence logic results in very compact, comprehensive and efficient implementations of large database applications. See Annex G. Conditions contain separate threads that navigate like creepers along the branches of the main data tree. This is very different from references by names in traditional languages. The conditions may result in deletions or copying. Copying is the main execution mechanism in Existence logic.”

The War of Universals

The Platonists claimed that universals only, like Persons, had a real existence. The universals were ideals, of which the singulars, like JOHN and MARY, were only imperfect shadows.

The Nominalists claimed that the universals were just flat words. Francis Bacon declared the Nominalists to be the winners of the conflict. With concepts and ontologies, the computer scientists of today seem to be back into the Dark ages.

In the Conceptual schema report from ISO, 1985, plurals, like Persons, are treated as sets. The individuals, like JOHN and MARY, are treated as members of this set. The singular indefinite form, person, is treated as a variable ranging over the elements of the set Persons. Any reader of ontologies should read on Interpreted Predicate Calculus in this report. This is a clean application of Higher order logic; which I do not subscribe to.

Ontologies, like DOLCE, seem to define the singular indefinite forms, which they call universals. They seem to define classes that appear in a (database) schema. The classes act as templates/prototypes for the instances. See about classes and instances in Existence logic, in the subsequent section.

The words universals, singulars, sets, individuals, instances, entities, classes, variables and types have been invented during different time periods and may each have several meanings.

Classes

In Existence logic, we distinguish between populations and schemata. A population node has a schema reference, S<>, to a schema node. The schema node may have a population reference to its population nodes.

The contents of schemata are called classes. The contents of populations are called instances.

Nothing is schema or population in an absolute sense; they are only so relative to each other. Likewise for classes and instances. Instances are copies of their classes. Existence logic is based on duplication. Hence, it uses significant duplicates. The mapping from instances to classes is homomorphic.

We use the following notation:

:	item/pixel
Indentation	subordinate
Line shift next	
<>	condition
‘	superior
(	subordinate in the one-dimensional notation
,	next in the one-dimensional notation
S	schema

```
:                                               SCHEMA 1
      :                                         Country
            :                                   Inhabitant
                  :                             Loved-person
                        <>                      ‘ ‘ (Inhabitant
:                                               POPULATION 1
      S
            <>                                  S ‘ ‘ (SCHEMA 1
      :                                         Country
            :                                   Inhabitant
                  :                             Loved-person
                        <>                      ‘ ‘ (Inhabitant, Inhabitant
                  :                             Loved-person
                        <>                      ‘ ‘ (Inhabitant, Inhabitant, Inhabitant
            :                                   Inhabitant
            :                                   Inhabitant
```

In the Figure, we see that there are many Inhabitant-s both in the population and its schema. The inscriptions Inhabitant do not refer to the same string.

Taxonomy

‘Figure 2: Taxonomy of DOLCE basic categories’ depicts a tree. The nodes are ‘categories’: I see no text on the meaning of the edges. Do they show sub-classes or containments?

The top is Particular; its subordinates are Endurant, Perdurant, Quality and Abstract.

I do not see the convenience of distinguishing Endurant and Perdurant. 'Endurant are wholy present; Perdurants 'happen in time'. The distinction deals only with time scales. Hence, a blink lasting a millisecond is long lasting, compared to an elementary particle lasting only nanoseconds.

I believe that Endurant, Perdurant and Particular should be collapsed into Entity, and an Entity may contain entities recursively.

I believe that Quality means Property, ie. Attribute and Attribute group. An Entity may contain Attribute-s or Attribute group-s, and Attribute groups may contain Attribute groups recursively or Attributes. Attributes may contain Values. Maybe, the Regions under Abstract mean value types.

Abstract has the subordinates Fact, Set and Region. I can accept that Set is an abstract, ie. a concept. I know Fact to mean a statement about phenomena that really are; to me, this is the opposite of Abstract. Region and its details make no sense to me.

PS: the categorization of data/phenomena into Entity, Attribute group, Attribute and Value is not fundamental, but may be practical.

PS: I have access to two pages only of the DOLCE document, and I do not have the axioms. I do not think they are relevant.

I conclude that Figure 2 does neither depict an appropriate class tree nor a file tree. I recommend to scrap it all.

Denotations

The statements about the phenomena are inside the observer, before they are uttered, and after they have been received. We find the contents of each statement, as they are sent or received, in a Contents population. A normalized version of each statement is found in an External terminology population.

There is a mapping between statements in the External terminology population and the phenomena inside an observer. We require that this mapping is isomorphic. This means that we require an isomorphic mapping between two syntactical structures. We require that this mapping is explicitly stated between each term and each phenomenon. However, we do not require that the mapping is onto, either way.

The following Figure shows an example population of normalized instances in what we call an External Terminology Population, and their mappings to Phenomena instances, inside an observer automaton. The schemata with classes are not shown.

The Figure is about a Country with the Name NORWAY, having three Inhabitant-s. The Inhabitant with the Name JOHN has two Loved-person-s, who each refers to an Inhabitant in the same Country. The Loved-person-s may have identifiers and attributes which are different from those of the referenced Inhabitant-s.

The Country, the Inhabitant-s and the Loved-persons have denotation mappings to Phenomena. The Names and values do not have denotations. We have not attached name tags to the phenomena, but we could have.

We use the following notation:

```
:               item/pixel
Indentation     subordinate
Line shift next
<>              condition
‘               superior
(               subordinate in the one-dimensional notation
,               next in the one-dimensional notation
D               denotation
```

```
:                                       EXTERNAL TERMINOLOGY POPULATION
        D <>                            ‘ ‘ (PHENOMENA

        :                               Country
                :                       Name (NORWAY
                D <>                    ‘ ‘ ‘ (PHENOMENA (:
                :                       Inhabitant
                        :               Name (JOHN
                        D <>            ‘ ‘ ‘ ‘ (PHENOMENA (: (:
                        :               Loved-person
                                D <>    ‘ ‘ ‘ ‘ ‘ (PHENOMENA (: (: (:, :
                                <>      ‘ ‘Country (Inhabitant (Name (MARY
                        :               Loved-person
                                D <>    ‘ ‘ ‘ ‘ ‘ (PHENOMENA (: (: (:, :, :
                                <>      ‘ ‘Country (Inhabitant (Name (LILL
                :                       Inhabitant
                        :               Name (MARY
                        D <>            ‘ ‘ ‘ ‘ (PHENOMENA (: (:, :
                :                       Inhabitant
                        :               Name (LILL
                        D <>            ‘ ‘ ‘ ‘ (PHENOMENA (: (:, :, :

:                                       PHENOMENA
        :
                :
                        :
                                <>      ‘ ‘: (:, :
                        :
                                <>      ‘ ‘: (:, :, :
                :
                :
```

Note that for instances having denotations, their classes will have to have denotations, as well. This also means that the phenomena will need to have classes. And, they are all data inside some observer. Hence, our nominalism is much more extreme than that of the Nominalists during the War of Universals.

In the next Figure, we have attached Norwegian name tags to the phenomena in a smaller example. We observe that the denotation mapping states nothing more than synonymity mappings. Sometimes this is useful; most often it is not useful.

:						EXTERNAL TERMINOLOGY POPULATION
	D <>					‘ ‘ (PHENOMENA
	:					Country
		:				Name
			:			NORWAY
				D <>		‘ ‘ ‘ ‘ ‘ (PHENOMENA (Land (Navn (NORGE
			D <>			‘ ‘ ‘ ‘ (PHENOMENA (Land (Navn
		D <>				‘ ‘ ‘ (PHENOMENA (Land
		:				Inhabitant
			:			Name
				:		JOHN
					D <>	‘ ‘ ‘ ‘ ‘ ‘ (PHENOMENA (Land (Navn (NORGE), Innbygger (Navn (JON
				D <>		‘ ‘ ‘ ‘ ‘ (PHENOMENA (Land (Navn (NORGE), Innbygger (Navn
			D <>			‘ ‘ ‘ ‘ (PHENOMENA (Land (Innbygger

:				PHENOMENA
	:			Land
		:		Navn (NORGE
		:		Innbygger
			:	Navn (JON

Note that derived data by functions may be added both among the External Terminology data and among the Phenomena.

Conclusion

The phenomena are data inside some observer.

The phenomena have the same kind of character as of other data. It makes no sense to develop a particular ontology, like set theory or mereology for phenomena.

The general theory of any phenomenon is the same as a minimal language for any data. This is the purpose of Existence logic.

The minimal language is a Grand Unified Theory of everything.