ACADEMIA | Letters

Attribute objects: An odd thing between objects and values

Manfred Jeusfeld

Classical data models such as the entity-relationship diagrams distinguish between objects (=entities) and values. An object is referenced by its immutable identifier that is assigned to the object when it is created. Its state is established by a combination of mutable values taken from so-called domains. Domains are sets of values that come with their own algebraic semantics, such as integer numbers or strings. So, values have no identity and do not change. Objects are identified and their state changes. In this paper, we challenge this strict dichotomy by introducing so-called attribute objects.

The knowledge representation language Telos [1,2] made its contribution to conceptual data modeling by consolidating all explicit facts (of a database) into a single data structure called propositions. When axiomatically formalizing O-Telos [3] as a variant of Telos, we deliberately excluded certain 'malformed' propositions. We now question this decision as it may have excluded an important category of facts, namely the attributes of objects, or more precisely distinguishing attributes from binary relations.

Attributes and relations in O-Telos

A proposition in O-Telos is a quadruple P(id,x,n,y) where id is the identifier of the proposition, x its source, n its name, and y its desitination. Propositions of the form P(x,x,n,x) are interpreted as nodes (links that start and end in themselves). All other propositions are relations. You may regard propositions as an extension of RDF triples where each triple is getting its own identifier. The axioms of O-Telos imply that node propositions are the only proposition, where the identifier and the source can be the same. Likewise, they are the only propositions where the identifier and the destination are the same. This forces to represent attributes of objects as relations, and ultimately to represent values as objects. Consider the object with label 'bill' that has two attributes age=27 and weight=72 and has a friend relation to the object 'mary':

P(o1,o1,*bill*,o1)

P(o2,o2,*mary*,o2)

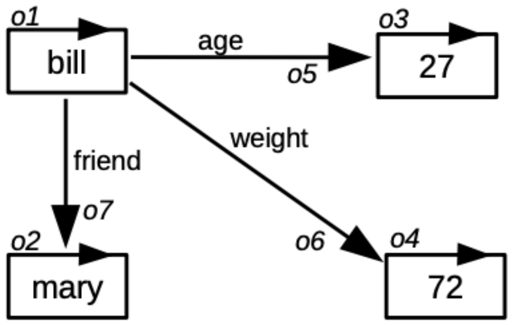
P(o3,o3,*27*,o3)

P(o4,o4,*72*,o4)

P(o5,o1,*age*,o3)

P(o6,o1,*weight*,o4)

P(o7,o1,*friend*,o2)



Hence in, O-Telos objects only have the immutable name as their state. Changes to the state are realized by removing and adding propositions. For example, the age of 'bill' can be changed by removing the proposition o5 and adding the propositions like P(o8,o8,*28*,o8) and P(o9,o1,*age*,o8). Note that 'attributes' such as o5 and o6 are represented in the same way as relations like o7. Indeed, O-Telos represents values as node objects such as o3 and o4.

But what happens if we lift the constraint on node propositions?

Q1) Do propositions, where the identifier and the destination are the same but not the source, make any sense?

Q2) Is there an interpretation for propositions where identifier and source are the same but not the destination?

For brevity, we only consider Q1 here. The answer to Q1 can be turned into an answer for Q2 because the two cases are virtually the dual to each other.

Attribute objects

An attribute object is a proposition of the form P(a,x,v,a) where ‘a’ is the identifier of the attribute object, ‘x’ its source (i.e. the object to which ‘a’ is an attribute), and v is the value of the attribute object. We assume here that ‘x’ is different from ‘a’, otherwise it would be just a node object like ’bill’.

With these attribute object we may rewrite the example:

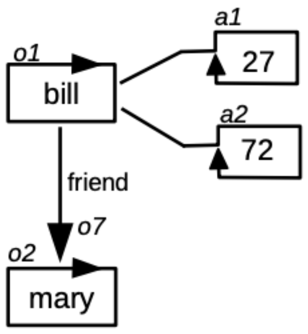
P(o1,o1,*bill*,o1)

P(o2,o2,*mary*,o2)

P(a1,o1,*27*,a1)

P(a2,o1,*72*,a2)

P(o7,o1,*friend*,o2)



You may note that we lost the place for attribute labels such as ’age’ and ’weight’. They can however be moved to the class level utilizing propositions of the form P(id,x,in,c), where ’in’ is a reserved label for declaring ‘x’ as instance of ‘c’. At the class level, we can declare age and weight as follows:

P(p1,p1,*Person*,p1)

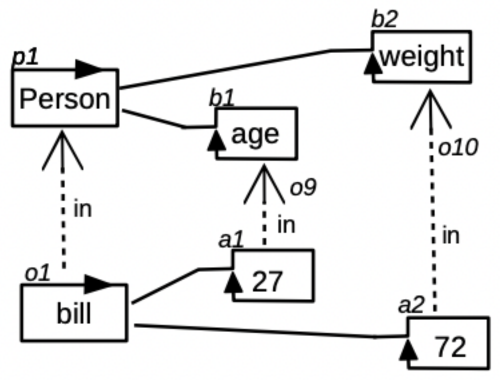
P(b1,p1,*age*,b1)

P(b2,p1,*weight*,b2)

P(o8,o1,*in*,p1)

P(o9,a1,*in*,b1)

P(o10,a2,*in*,b2)

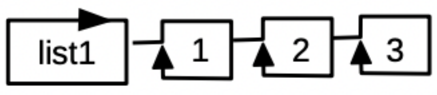


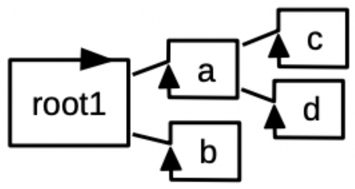
Hence, the attribute objects b1 and b2 are the classes of the attribute objects a1 and a2 and provide for the labels of the attributes, much like in UML class and object diagrams.

Data structures

The attribute objects require another object for their source, e.g. the object o1 (bill) is the source of the attribute object a1 (27). That is however all they need in contrast to the O-Telos attributes and relations that require to declare the attribute values such as 27 as explicit objects.

A direct benefit of attribute objects is that they can be chained, e.g. to form lists and trees:





For the brevity of this paper, we do not include the type level of these data structures. It would look similar to type/class level in the previous figure by defining integer and string attribute types to form lists and trees.

Triples or Quadruples?

The Telos propositions quadruples have been compared to RDF triples [4]. The main difference is that Telos propositions assign identifiers to all explicit facts, enabling thus their reification. RDF falls short of this ability and there seems to be no easy way to transfer the idea of attribute objects directly to RDF. Note however that attribute objects P(a,x,n,a) have a redundant occurrence of the parameter 'a'. Hence the triple T(x,a,n) contains the same information. The parameter 'a' is an identifier. It can be translated to a readable label by pulling it from the class of 'a'. But this leaves the ability to form data structures an open problem.

Does it matter?

The attribute objects introduced above are strictly derived from O-Telos 'proposition' quadruples by lifting a constraint on their structure. At least they show that there is an interpretation for such facts. But there is more about them. They allow us to strictly distinguish relations between objects such as the 'friend' relation from attributes whose values are no longer objects.

Further, by chaining attribute objects, we can describe complex data structures. The representation as proposition facts allow to follow the links between attribute objects in such chains in both directions.

Attributes objects also allow to reuse the same label for different attribute objects. For example, there may be an attribute object P(a3,o2,27,a3) to denote the age of 'mary'. This reminds of two memory cells that happen two have the same value stored in them. Again, this is closer to the semantics of UML class and object diagrams, and to the von Neuman memory model. In Telos, a value like would be represented as a node-like object P(o3,o3,27,o3). The axioms of Telos exclude another object with the same label.

The data structures of lists and trees can be declared and typed at the class level, e.g. for lists of integer numbers. Domains such as integer and string require another layer of definitions. We omit it here for brevity.

In essence, we have shown that attribute values do not need to have object identity in a language such as O-Telos that is based on a single quadruples fact table of propositions to represent objects, classes, relations, and attributes. Quadruples are indeed a more expressive mechanism than RDF triples.

The dual concept to attribute objects are propositions of the form P(a,a,n,x). They have themselves as source and point to another object x. Interestingly, they can be combined to form a self-referential pair of objects such as P(a,a,n,b), P(b,a,m,b).

References

1. J. Mylopoulos et al. (1990) Telos: Representing Knowledge About Information Systems. ACM Trans. Inf. Syst. 8(4): 325-362.

2. M. Koubarakis et al. (2020) A retrospective on Telos as a metamodeling language for requirements engineering, Requirements Engineering, 2020.

3. M. Jeusfeld (2005). Complete List of O-Telos Axioms. Online: http://merkur.informatik.rwth-aachen.de/pub/bscw.cgi/d1228997/O-Telos-Axioms.pdf.

4. M. Wolpers et al. (2002) An O-Telos Provider Peer for the RDF-Based Edutella P2P-Network. AP2PC 2002: 150-157.