Discussion on the ARM Phase/State Proposal

So, we could do something like the above but I'm afraid it's very ripe for poor semantics, especially with regards to our proposed solution to follow the model of E13 and the typing. I am still pro some solution of this sort, but this document outlines some concerns.

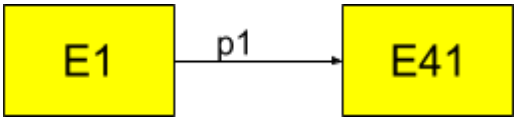
Let me explain why I think following the E13 solution will create problems, and then make a tentative counter proposal (which likely also has problems).

First let's identify kinds of things we want to say.

Sometimes we will want to make a time indeterminate statement. I will take our favourite example of naming.

1. Time Indeterminate Statement

We can already make time indeterminate statements about naming:



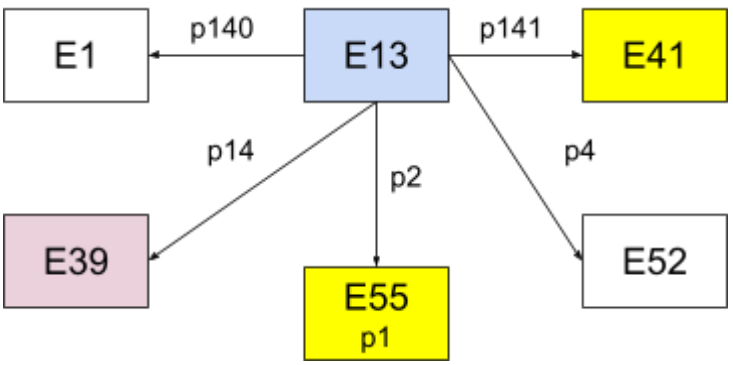
This says " Something is/was called by some name".

We do not know when this came to be or if it still the case. Since we state it now, it seems that it has not stopped. The information is stated as if it were ever valid.

This is a perfectly valid level of knowledge, but it doesn't allow us to say anything about time. It actually does tell us about a state of affairs (time indeterminately)

2. Event Initiated Time Determinate Statement

Using CRM logic, we can also make an event initiated time determinate statement about naming. The graph of that looks like this:



This says: “Something was called ‘something’ by somebody at some time.”

Here what we express is that there was an event about which we can know the actors and the time (inter alia) and it caused some state of affairs to come into being, namely, the having such and such a name of some thing.

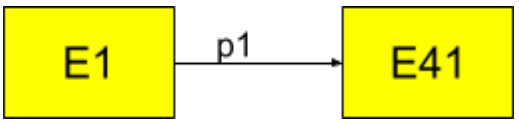
This allows us to express the kind of knowledge where we are aware of an intentinoal event of the kind described by E13 that makes it be the case that some property comes to hold between two things.

It is noteworthy to say, moreover, that we use a sort of meta conventional reference to achieve this expressiveness, since we refer to the CRM properties themselves as instances of the E55 Type class.

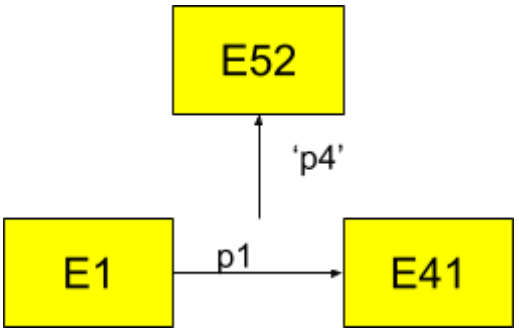
3. State Time Determinate Statement

We also want to make it possible to be able to express a different kind of knowledge regarding some property holding between two things at some point in time. We want to be able to express that some property held/holds between two things without referencing the initiating or terminating events that brought this to be because we do not know them and/or is not relevant to our research. (other motivations for state?).

We basically want to say:



But we want to limit it so that it says:

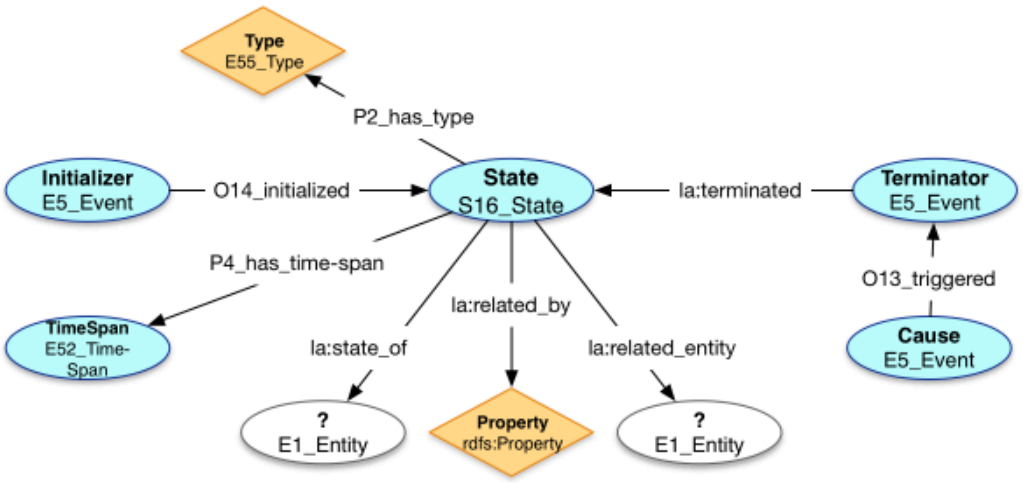


The above diagram is illegal in CIDOC CRM syntax, but would mean,

“ Something was called by ‘some name’ at some time”.

To do this properly, we/ARM now discuss introducing a state or a phase class (or using a pre existing one). The state or phase is the condition of the holding of one (or more?) properties between by one entity in relation to another one in the CRM world (what is already expressed by a property but without time). This holding of a state of affairs is just like a property statement (e.g.: E1->p1->E41) but we have additional knowledge that the time duration for which this statement was valid is limited (not by start and end events of which, the argument goes, we are ignorant).

In order to try to capture this with the existing semantic structures of CIDOC CRM (and CRMsci) we have sketched this:



Discussion:

1. The traditional CRM SIG objection against States I do not think is a problem here.

   A main objection is that it is not possible to identify states reliably, so it breaks integration. But it seems to me that that we already have properties in CRM and properties just do indicate states. Moreover, properties have scope notes and scope notes enable giving a way to identify an instance. Therefore, I would argue states are in principle identifiable.

   Anecdotally on the naming example, we have the state of ‘having a name’. Well ‘having a name’ holds just in case there exists some actor that attributes that name to some thing.

   “We used to call it ‘bazingo’, ca. 1950, but they don’t call is that anymore.”

   This seems like it could be usefully represented by our proposed state class. It is known that something had a name for such and such a time, but not what made it happen nor what made it stop.

   We can recognize empirically instances of ‘having a name’.

2. The problem with State

I still think there is a problem with our proposed solution as documented in the proposed diagram (see above).

Particular I think that this solution (referencing all CRM properties from our state class) allows ontological nonsense to be stated (a problem that does not occur with E13).

If we allow the ‘related by’ property to point to any property then we potentially introduce a bunch of non-sense ontologically and we lack the ability to more definitely define the kind of state/phase which we are talking about (which arguably will be very important to the user)

What is the non-sense? Well, it occurs to me that ‘states of affair’ should be about persistent items (perdurants) and not about temporal entites (endurants). Why? Well, although perdurants and endurants are ultimately a bit of a fiction, they are functional distinctions. A perdurant is supposed to have static, stable relations over time unless temporal entities bring it into relation with other endurants such that it is caused to change. Temporal entities model flux/motion/process and if we need to point to subparts of this, we just do subdivision of the process (the example of modelling the orchestra with a bunch of individual performances within… it would not make sense to time limit the properties within the performance to indicate that it only happened for such and such a time)

Our Phase/State class, I would argue should then only have to do with properties that hold between an endurant and an endurant. What we want to enable saying is that, some endurant had different substantial relations with another endurant for some time span, but I am ignorant of events that made that be the case (or nothing made it be the case, it just held at that moment).

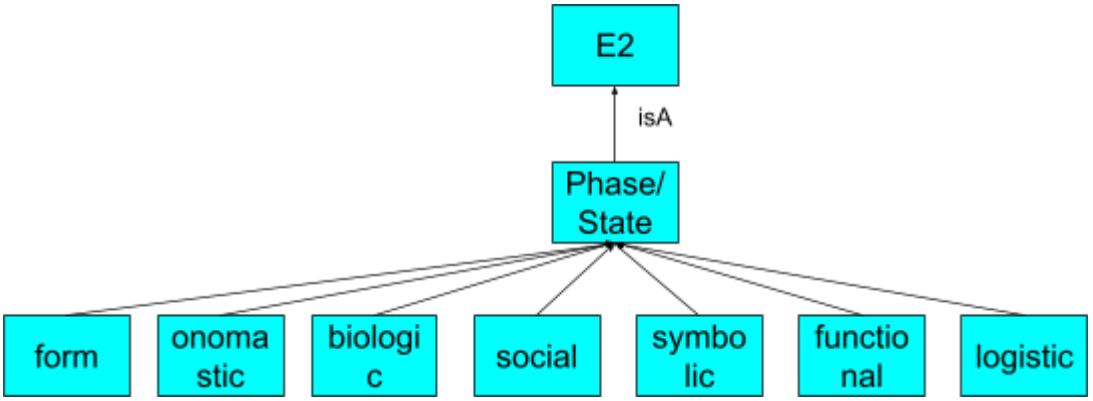
It should not, in my opinion, be used to talk about relations between perdurants and endurants because we can model that via the event model.

If the above arguments hold and others agree with the logic then:

In order to stop non-sense from being formulated through an ARM phase/state class, I would argue rather to do a more fine grained modelling of phase/state of the following type.

We look at the collected properties that hold between perdurants in CRM and we understand what function they hold and we create a specific phase class for them. We limit the la:related_by property for this subclass to a certain bundle of extant crm properties that are logically coherent and consistent. We create a container

phase/state class that allows you to query over all of these. It would look something like this:



Onomatic: bundles CRM properties that have to do with an endurant holding a name
Biologic: no examples, just thought I would throw it in there

Social: bundles CRM properties have to do with holding some sort of social relation like owning

Symbolic: bundles CRM relations that have to do with representation

Functional: bundles relations for perdurants as with regards to their use

Logistic: bundles relations for location that require time

Form: size, shape, orientation relations perhaps

I admit that this may look like a beast and I am sure there are errors, but I generate it in order to try to address issue number 2 above, which I believe is quite serious. For E13 it is okay to allow all properties of CRM to be referenced. For State I am not sure that it is ontologically correct.

Something ‘like’ the above would allow us to group kinds of state at least (and they would differ in substance and be identifiable)...