Victor Morgante
May 8, 2020 · 13 min read · · Listen

Ambiguity and ORM2 (Object-Role Modeling v2)

ORM2 is a type of fact-based modelling

ORM2 (Object-Role Modeling v2) is a type of fact-based modeling, which is a type of concept
Role Modeling is a graphical language.

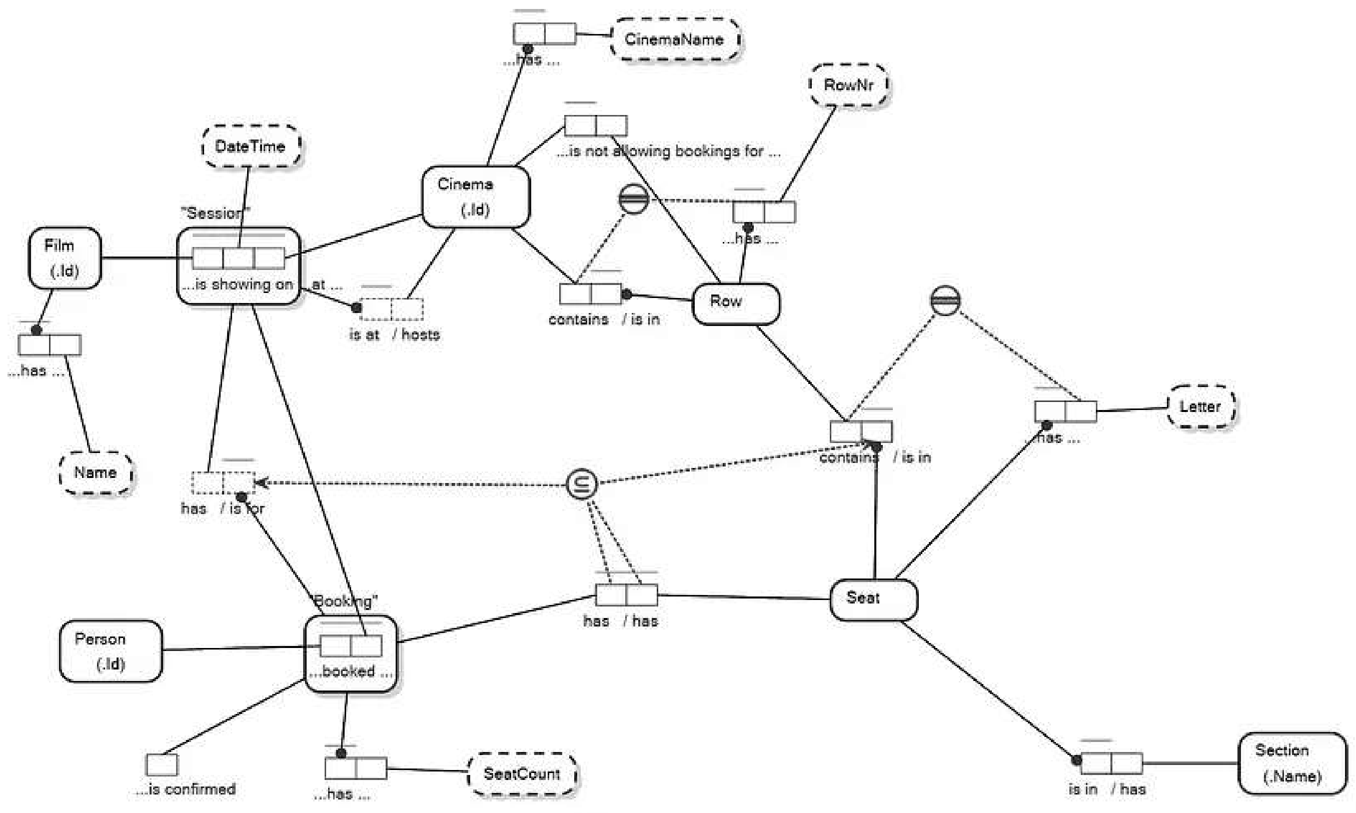
The short answer to the question posed in the title of this article is:

“Yes, under certain circumstances, sentences of ORM2 can be constructed, the interpretati are ambiguous”.

This might be contrary to marketing material you have read on Object-Role Modeling, or indeed contrary to academic papers published on Object-Role Modeling and by journals of varying reputation.

Things are not entirely lost for the application or utility of the ORM notation, only to any claim to the efficacy of ORM under the formalisation of ORM. It would be dishonest for my company to produce ORM based software without having first disclosed the limitations of ORM outside of software.

Consider the following ORM2 diagram:



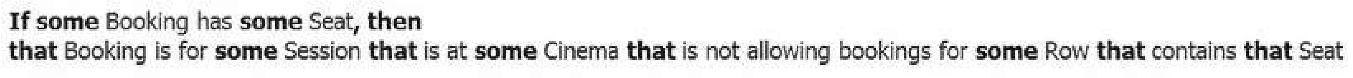
The symbol in the middle of the diagram is a Join Subset Constraint. It looks like this:

⊆

Subset Constraint Symbol in ORM

A Join Subset Constraint joins a set of roles played by ostensible objects in a subset / superset relationship.

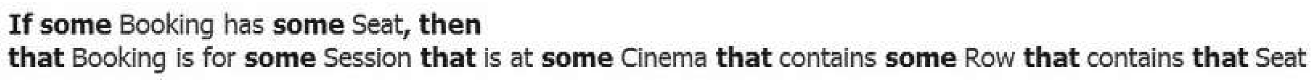
From the model above, it is ambiguous as to whether the Join Subset Constraint says which of the following:



If some Booking has **some** Seat**, then**
that Booking is for **some** Session **that** is at **some** Cinema **that** is not allowing bookings for **some** Row **that** contains **that** Seat

Verbalisation of a Joinn Subset Constraint

or this....



If some Booking has **some** Seat**, then**
that Booking is for **some** Session **that** is at **some** Cinema **that** contains **some** Row **that** contains **that** Seat

So basically, an interpreter of that diagram would not be able to distinguish as to whether a booking was made for a seat in a row in a cinema where that ***was not allowed by the cinema***, or whether the booking was made for a seat in a row at a cinema that merely ***contains that row***.

NB See the bottom of this article where we cover a case where we say that Fact Type Readings cannot have negation.

Clearly this is not an ideal situation for a visual language promoted as formalised, as in logically formalised?

To understand that question/statement we need to dive into mathematics and logic. In mathematics/logic a language or a theory is said to be formalised if there is a well defined set of rules as to how sentences in that language are constructed. i.e. You can algorithmically determine as to whether sentences in that language are *well formed* or not. If a sentence constructed by someone does not meet those rules, then it is said to not fit within the formalisation of that language.

So, you might ask, if a language is formalised then surely it cannot be ambiguous? Well, that's not true. You see, there is the formalisation and then there is the interpretation of sentences that are well formed in that language. In layman terms, although sentences in English (say) may be grammatically well formed, the sentence may otherwise be misinterpreted depending on how you interpret the meaning of the words in the sentence. This is well known for spoken languages, and is the underpinning of formal logic. More specifically formal logic by way of symbolic logic was synthesised with the aim of actually reducing ambiguity to nothing under a correctly synthesised theory.

So, again, does that make ORM2 unreliable as a language? Can I just send someone an ORM diagram and expect them to understand what I am saying even if their interpretation of ORM as a language is exactly the same as mine?

The answer is that in its written form and under examples like what I have provided here and for join subset constraints, "**Yes. ORM2 allows for ambiguous sentences. No, you cannot just send someone an ORM2 diagram with a Join Subset Constraint constructed as in the example above and expect the diagram to be interpreted exactly as you intended**". Not outside software that is. You could not send them a piece of paper with the diagram above and expect no ambiguity.

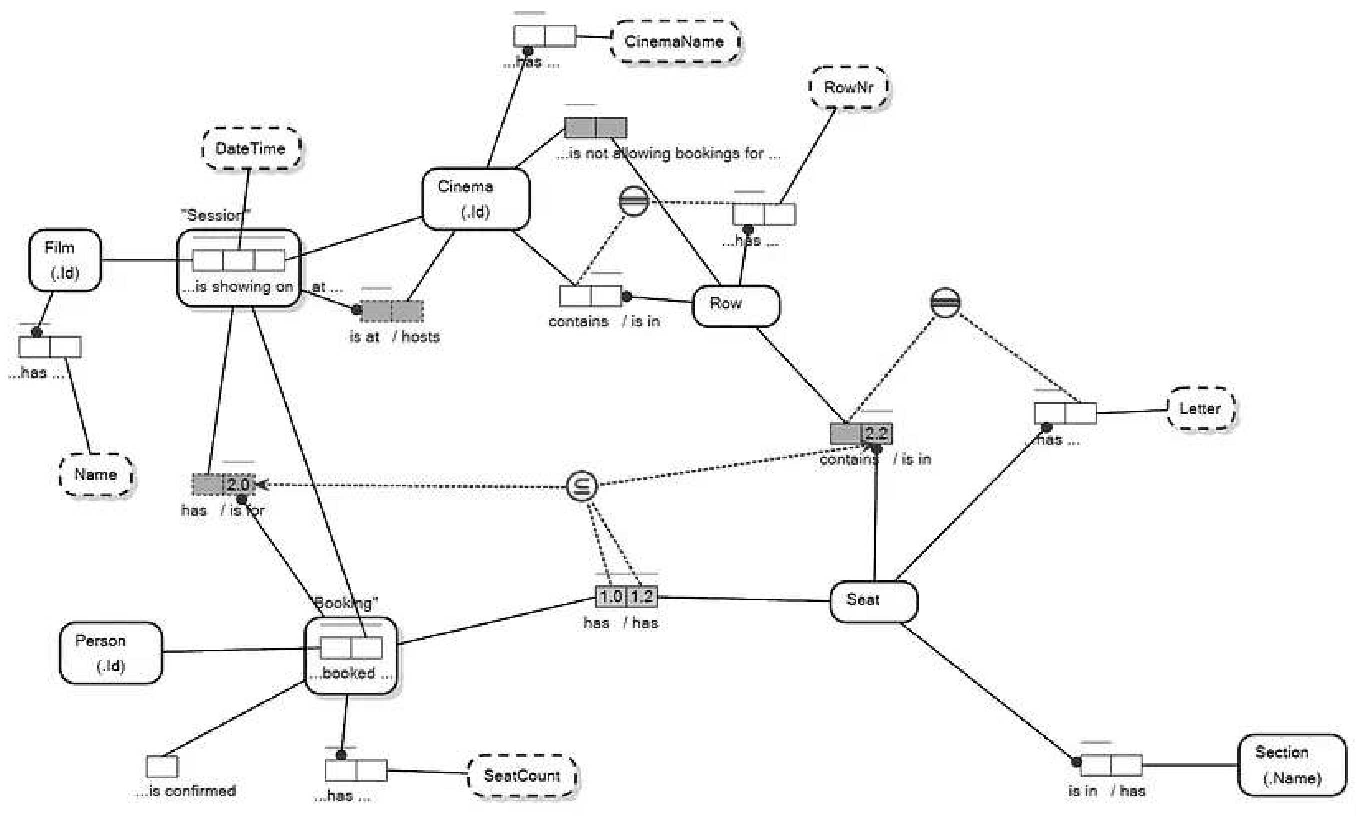
In logical terms, ORM2 may well be formalised, but the theory has an interpretation which contains ambiguity, so the formalisation of ORM2 means very little. With join subset constraints, the formalisation of ORM2 is pointless from a logical perspective.

Update 2020/09/13: More succinctly, a set of well formed sentences of ORM may be constructed, the interpretation of which is ambiguous. While true of any theory of logic, a subset join constraint formed in the manner above, and in relation to the overall ORM diagram, cannot be mapped to one set of theorems of KL (described below), but two…breaking the isomorphism.

You may think it a risky proposition for someone who has spent years developing an ORM based software tool to expose the weaknesses of ORM. The way I see it, it would be riskier not to expose the weaknesses of ORM. As a customer, you have to know what you are buying. But things are somewhat salvageable under software implementations of ORM, and I will explain.

In software, you get to explicitly define the join path of Join Subset Constraints. i.e. Rather than the join path being implicit, in a formal logic sense, you get to explicitly define the join path (as below). What this means for Dr Halpin's PhD thesis, underpinning the original ORM, I will then explain further.

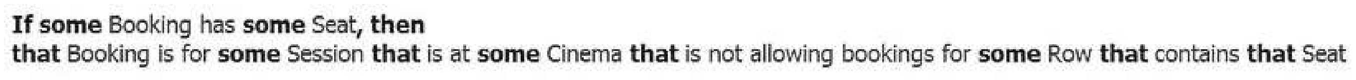
Consider the following diagram:



Explicitly defined join path in Object-Role Modeling software

The diagram above was generated in our Boston Object-Role Modeling software. The light blue coloured roles represent the *subset* component of our join subset constraint. The purple roles represent the path of the *superset* component of our join subset constraint.

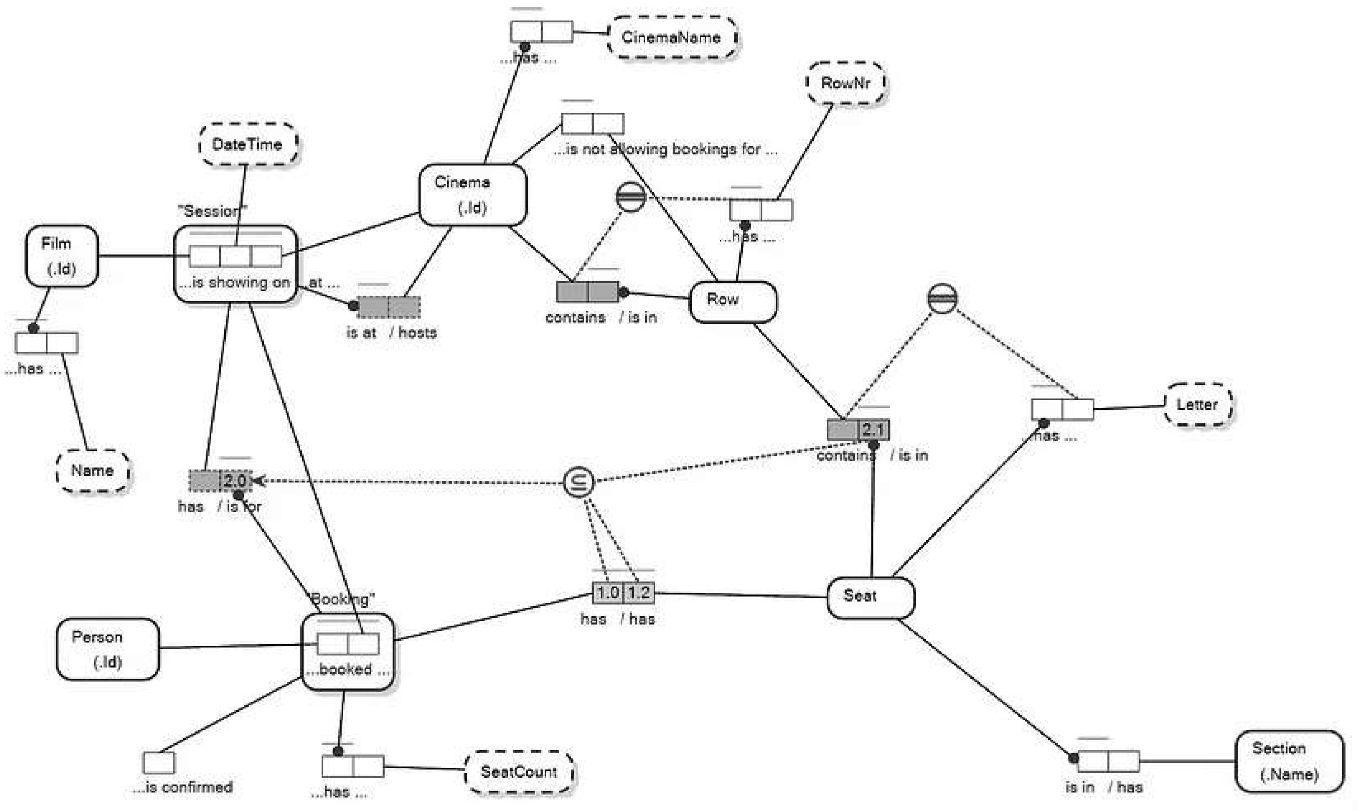
It is easy, in this regard to clearly interpret the model as reading:



If some Booking has **some** Seat**, then**
that Booking is for **some** Session **that** is at **some** Cinema **that** is not allowing bookings for **some** Row **that** contains **that** Seat

Verbalisation of a Join Subset Constraint in Object-Role Modeling i.e. the *join path* of the Join Subset Constraint is explicitly defined in the software and when you click on the join Subset constraint you get to see the explicit path. **Just looking at the diagram without clicking on the join subset constraint symbol to see the explicit path would lead you to conclude that ORM2 is an ambiguous language; as indeed it can be.**

NB Of course, the same example could be made for the alternative interpretation of our subset join path example, as below:



Explicitly defined join path of a Join Subset Constraint in ORM

...which reads for the join subset constraint...

If some Booking has **some** Seat**, then**
that Booking is for **some** Session **that** is at **some** Cinema **that** contains **some** Row **that** contains **that** Seat

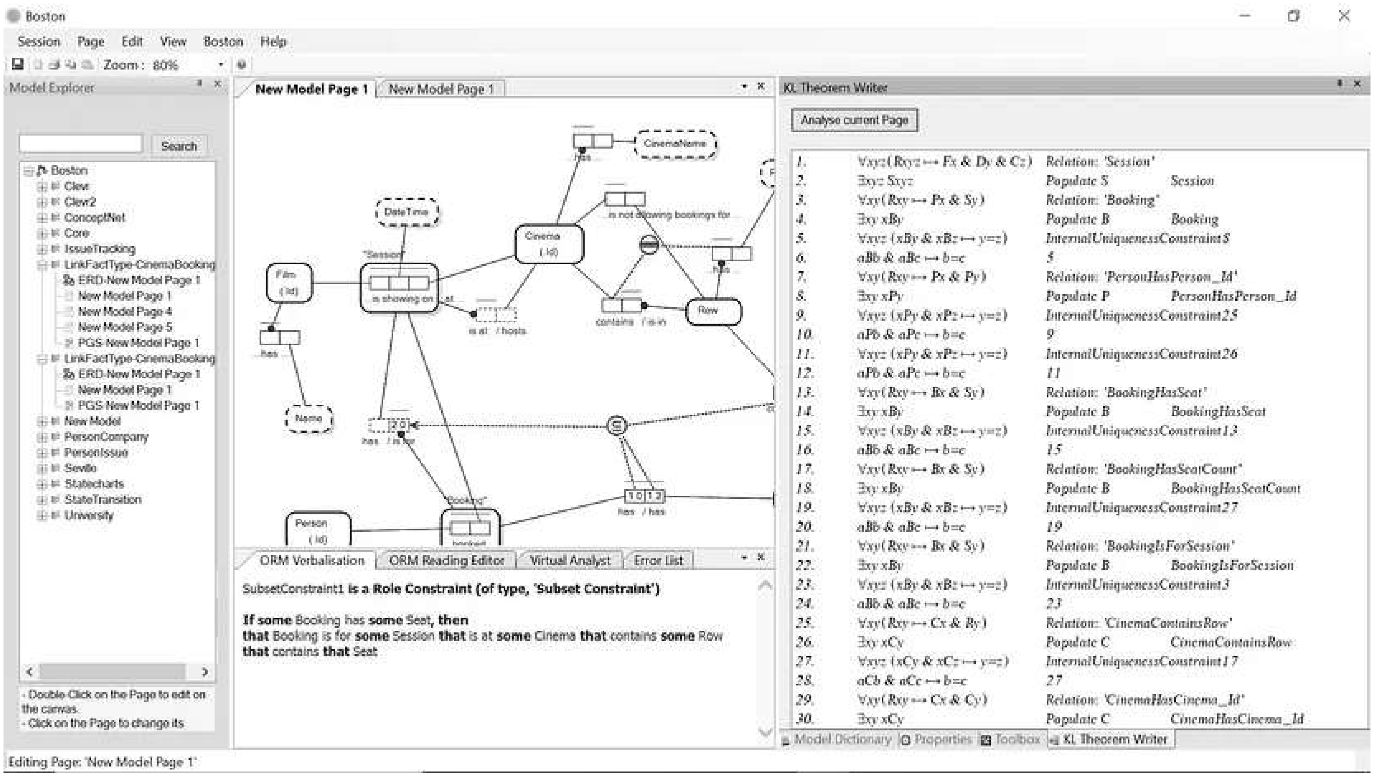
What does this say about Object-Role Modeling and its Formalisation?

As we have just seen, even though sentences of a formalised language are well formed, that does not mean they are not ambiguous.

Much marketing material and many academic papers submitted on Object-Role Modeling take pains to emphasise that ORM is *formalised*, pointing to the PhD thesis of Dr Terry Halpin who first formalised the predecessor to ORM2 (in which Join Subset Constraints was released), called Niam, and as accepted in 1989.

The basic thesis of Dr Halpin's doctoral dissertation was that as a graphical language Niam (or ORM1 lets call it) is isomorphic with a theory of symbolic logic synthesised under finite model theory, called KL or Knowledge Language. The thesis was that if theories (languages) of symbolic logic are constructed in such a way that they have a standard interpretation that is not ambiguous, and that symbolic language has a one-to-one mapping with an otherwise graphical language (like Niam) then you can say that that graphical language is also unambiguous.

Examples of the symbolic KL language, side by side with an ORM diagram is shown in the picture below:



```
1.   ∀xyz(Rxyz ↦ Fx & Dy & Cz)   Relation: 'Session'
2.   ∃xyz Sxyz                   Populate S     Session
3.   ∀xy(Rxy ↦ Px & Sy)          Relation: 'Booking'
4.   ∃xy xBy                     Populate B     Booking
5.   ∀xyz (xBy & xBz ↦ y=z)      InternalUniquenessConstraint5
6.   aBb & aBc ↦ b=c             5
7.   ∀xy(Rxy ↦ Px & Py)          Relation: 'PersonHasPerson_Id'
8.   ∃xy xPy                     Populate P     PersonHasPerson_Id
9.   ∀xyz (xPy & xPz ↦ y=z)      InternalUniquenessConstraint25
10.  aPb & aPc ↦ b=c             9
11.  ∀xyz (xPy & xPz ↦ y=z)      InternalUniquenessConstraint26
12.  aPb & aPc ↦ b=c             11
13.  ∀xy(Rxy ↦ Bx & Sy)          Relation: 'BookingHasSeat'
14.  ∃xy xBy                     Populate B     BookingHasSeat
15.  ∀xyz (xBy & xBz ↦ y=z)      InternalUniquenessConstraint13
16.  aBb & aBc ↦ b=c             15
17.  ∀xy(Rxy ↦ Bx & Sy)          Relation: 'BookingHasSeatCount'
18.  ∃xy xBy                     Populate B     BookingHasSeatCount
19.  ∀xyz (xBy & xBz ↦ y=z)      InternalUniquenessConstraint27
20.  aBb & aBc ↦ b=c             19
21.  ∀xy(Rxy ↦ Bx & Sy)          Relation: 'BookingIsForSession'
22.  ∃xy xBy                     Populate B     BookingIsForSession
23.  ∀xyz (xBy & xBz ↦ y=z)      InternalUniquenessConstraint3
24.  aBb & aBc ↦ b=c             23
25.  ∀xy(Rxy ↦ Cx & Ry)          Relation: 'CinemaContainsRow'
26.  ∃xy xCy                     Populate C     CinemaContainsRow
27.  ∀xyz (xCy & xCz ↦ y=z)      InternalUniquenessConstraint17
28.  aCb & aCc ↦ b=c             27
29.  ∀xy(Rxy ↦ Cx & Cy)          Relation: 'CinemaHasCinema_Id'
30.  ∃xy xCy                     Populate C     CinemaHasCinema_Id
```

SubsetConstraint1 **is a Role Constraint (of type, 'Subset Constraint')**

If some Booking has **some** Seat, **then**
that Booking is for **some** Session **that** is at **some** Cinema **that** contains **some** Row
that contains **that** Seat

Symbolic language KL side by side with a corresponding Object-Role Modeling diagram

To mathematicians and logicians, symbolic logic (as on the right in the picture above) is very important and garners lifetimes of research and analysis. The idea of Dr Halpin's thesis was to make that symbolic logic world more accessible to people who don't study symbolic logic but who do develop models for computer software. The thesis aimed to achieve this by mapping an isomorphism from a graphical language to a symbolic language such that the graphical language could be thought of a synonymous with the symbolic language, and vice versa.

In a further article I will discuss as to whether putting two languages side by side constitutes proof of isomorphic mapping, but for now let us say that Dr Halpin's thesis was accepted and for Niam at least a lot of good things can be said about its usefulness to the world. This, because Niam does ***not*** support join subset constraints.

For ORM2, with the introduction of Join Subset Constraints, things are a lot less clear and are, indeed, ambiguous.

What this means is that any paper published since the introduction of ORM2 circa 2005 and that cites/references Dr Halpin's PhD thesis as to the efficacy of ORM (while actually referring to ORM2) as a Formal Language, was published to the peril of the legitimacy of that paper if not otherwise disclosing the ambiguous characteristics of ORM2. You see, **Dr Halpin's PhD thesis would only hold water for ORM2 if you took a walk in the park, picked up a piece of paper with the diagram at the top of this article and tried to unambiguously interpret that diagram, you can't. Dr Halpin's thesis does not hold for ORM2.**

That means that papers hoping to garnish support as to the unambiguous nature of ORM2 by pointing to its formalisation, neglect that under formal logic formalisation is pointless without a standard interpretation that is unambiguous, otherwise you may be just wasting your time formalising sentences under that theory. Why bother formalising a language if its interpretation is ambiguous?

In a previous article, Fact-Based Modeling — An Apology, I argued that those who are belittled by ORM advocates for using competing graphical languages, such as Entity Relationship Diagrams or Property Graph Schemas or the languages of the UML, because they are otherwise not formalised under a PhD thesis, should not take that to heart. And should point to this article and say, "Well since 2005 Object-Role Modeling, now in its second iteration, is no longer unambiguous and Dr Halpin's thesis means little outside a software world where Join Subset Constraints can be explicitly defined". It may also be time to re-evaluate some papers that have been written since 2005 and accepted by otherwise reputable journals. We should at least be wary of papers claiming virtue of formalisation of a language.

It is important that mathematics and logic retain its integrity where possible. It is also important that journals and academic professionals strive for the best possible exposition of state of the art in areas of research.

This article does not aim to denigrate ORM2 or any one person, but it does intend to bring equity to the playing field and dissuade hype where possible. I feel it is important in a field otherwise imbued with marketing material that may otherwise misinform people.

Where does that leave ORM? In my view, ORM is useful when used in a software environment where it can be used without too much hype.

NB A new body of work attempts to formalise ORM2 in a similar effort to Dr Halpin's PhD thesis. The work is current in the year 2020. This time, no attempt to put ORM diagrams side by side with theories of symbolic logic, to imply an isomorphism, is made. The reader of the document is left to make their own conclusions, or accept on blind faith that sentences of the newly formalised language of symbolic logic is isomorphic with sentences of ORM2. The new document can be found here. The document explicitly invites the work to be referenced and cited. This may be the first citing. My concern is that there is a misinterpretation of the relationship between formalisation and interpretation, model theory, formal logic and the basis of isomorphism in this new work. One need only point to the example at the top of this article to show that ORM2 sentences can be ambiguous. Or, one can cite the work and point to how this new 'formalisation' does not change the fact that ORM2 sentences can be ambiguous.

Addendum — 2022-05-02 Halpin's confirmation of the capacity for ORM2 models to be ambiguous

Researchers, please viz the following:

> *"In the absence of further marks on the schema diagram, ORM join constraint paths may sometimes be ambiguous.", Halpin [1 p. 7]*

True

> *"In ORM, it was possible to encounter constraints involving role paths where the diagram is ambiguous as to which role paths are intended.", Halpin [2 p. 11]*

True, though this was not resolved in ORM2 as below.

> *"Currently ORM does not include any graphic notation to highlight the relevant joins (as we have done here using shading). In general, shading is not adequate since a path may traverse the same predicate more than once. A general graphical solution may be provided by numbering the roles used to perform the joins, but this can lead to messy diagrams", Halpin [3 p. 7]*

I.e. Halpin has made no effort to fomalise join paths in ORM2 because he has either decided the problem is undecidable or realises that shading/numbering of role join paths in software breaks away from Halpin's PhD thesis in that ORM diagrams with such ambiguous join paths cannot be disambiguated without software and if software is required then any language (e.g. the UML) could be considered 'formal' under Halpin's interpretation of the word because any bug free software (including UML software) has a proof based on a Curry Howard Correspondence. I.e. Voluminous claims by Halpin that ORM(2) is *formalised* become obsolete and vacuous. NIAM may have been formalised and may have been isomorphic with a theory under finite model theory (KL), but not ORM2.

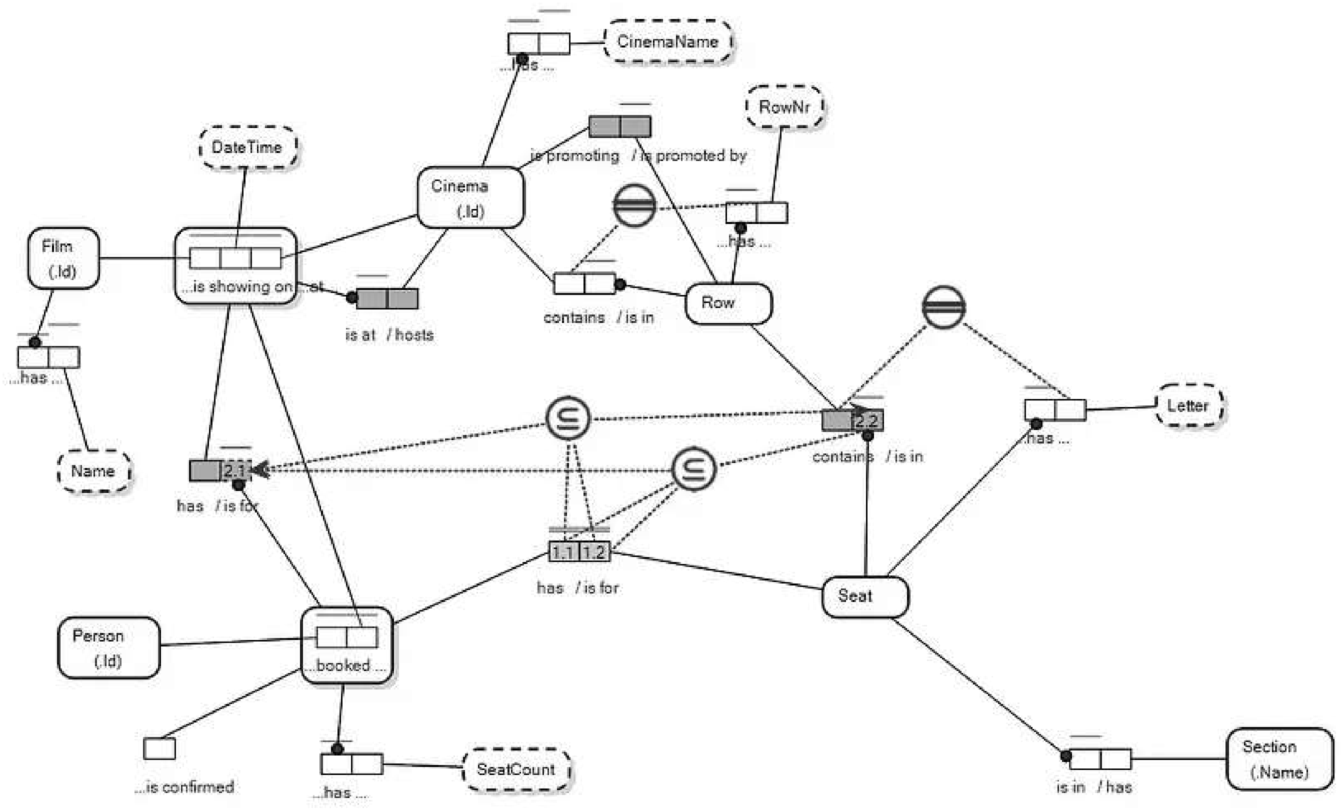
> *"ORM 2 removes that ambiguity by enabling the display of role-sequence numbers to be toggled on/off.", Halpin [2 p. 11]*

Invalidated by [3 p. 7] and as above. I.e. Toggling on/off requires software to disambiguate a diagram, and numbering of role paths doesn't help, as per the example below in this addendum.

> *This problem may easily be resolved by having the modeler indicate the path in some way (e.g. by shift-clicking the predicates on the path) and then displaying this path in some way when the constraint is inspected (e.g. by shading)", Halpin [1 p. 7].*

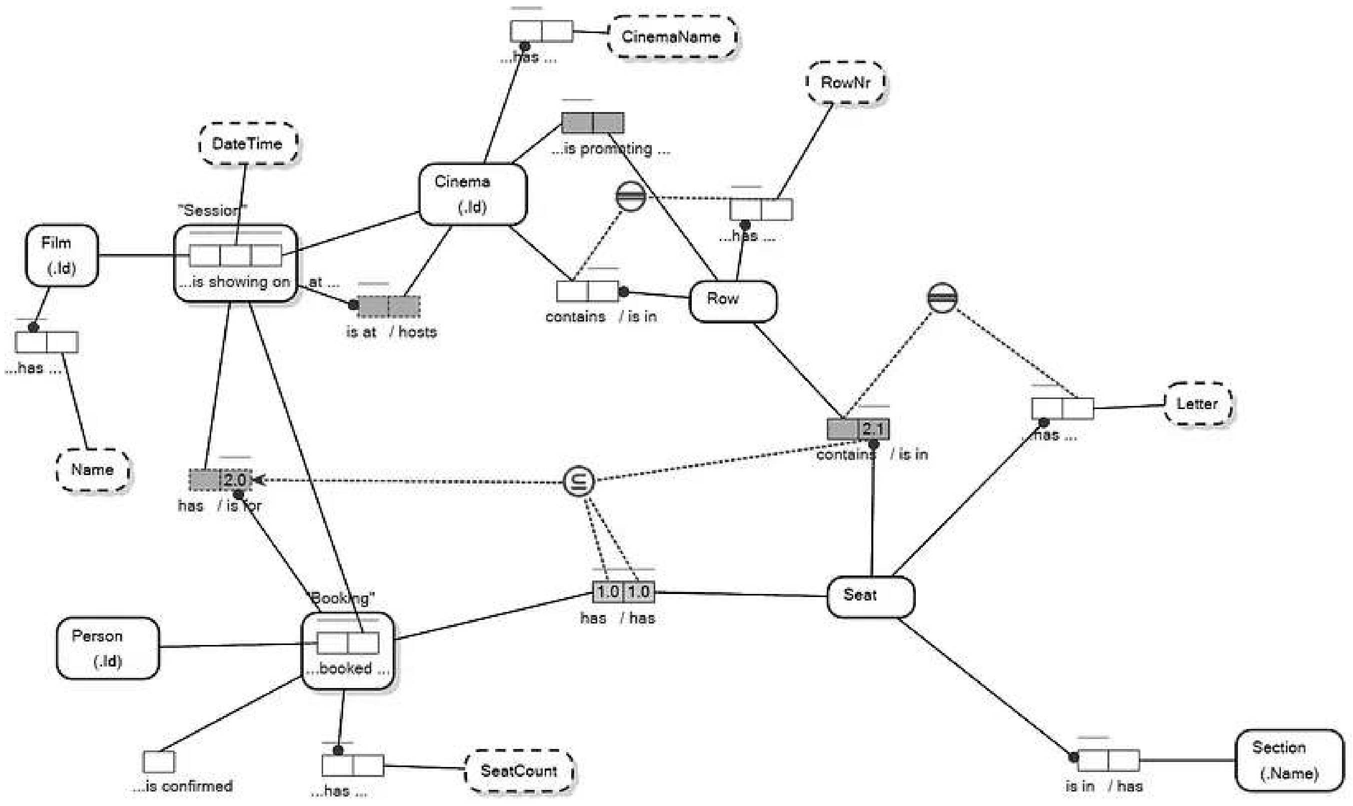
False, as below. Which of the two Join Subset Constraints is responsible for the shading and numbering? It is an ambiguous ORM diagram and only valid in ORM software. I.e. The software knows which Join Subset Constraint owns the join path but it is ambiguous as to a human looking at the diagram as to which. Halpin has made no effort to further formalise ORM2 to compensate for this ambiguity.

See also: A suitable counterexample



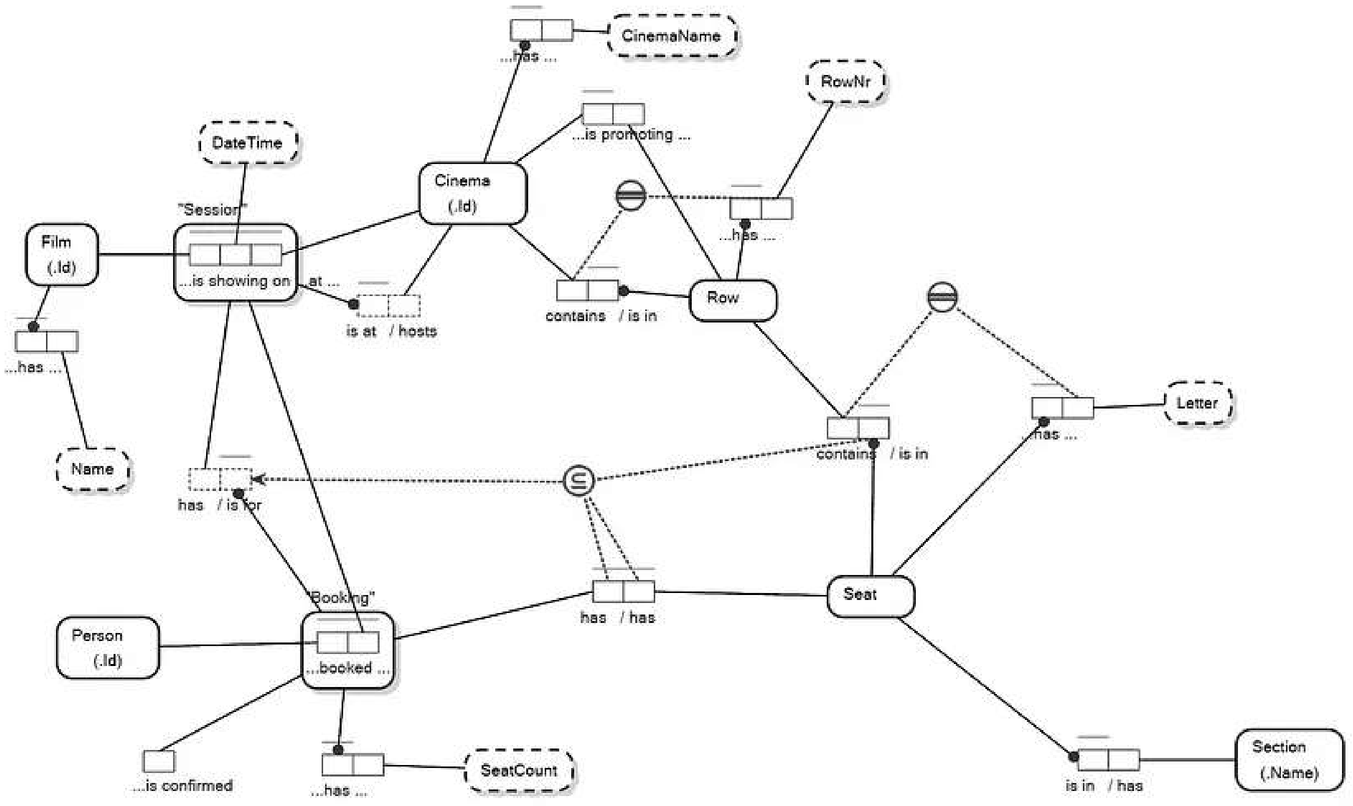
Addendum — Addressing where Fact Type Readings cannot contain negation

Let us say that Fact Type Readings in Object-Role Modeling, so that we do not allow us to say "Cinema is not allowing bookings for Row". We merely modify our example to include ambiguity over a different Fact Type Reading; as below:



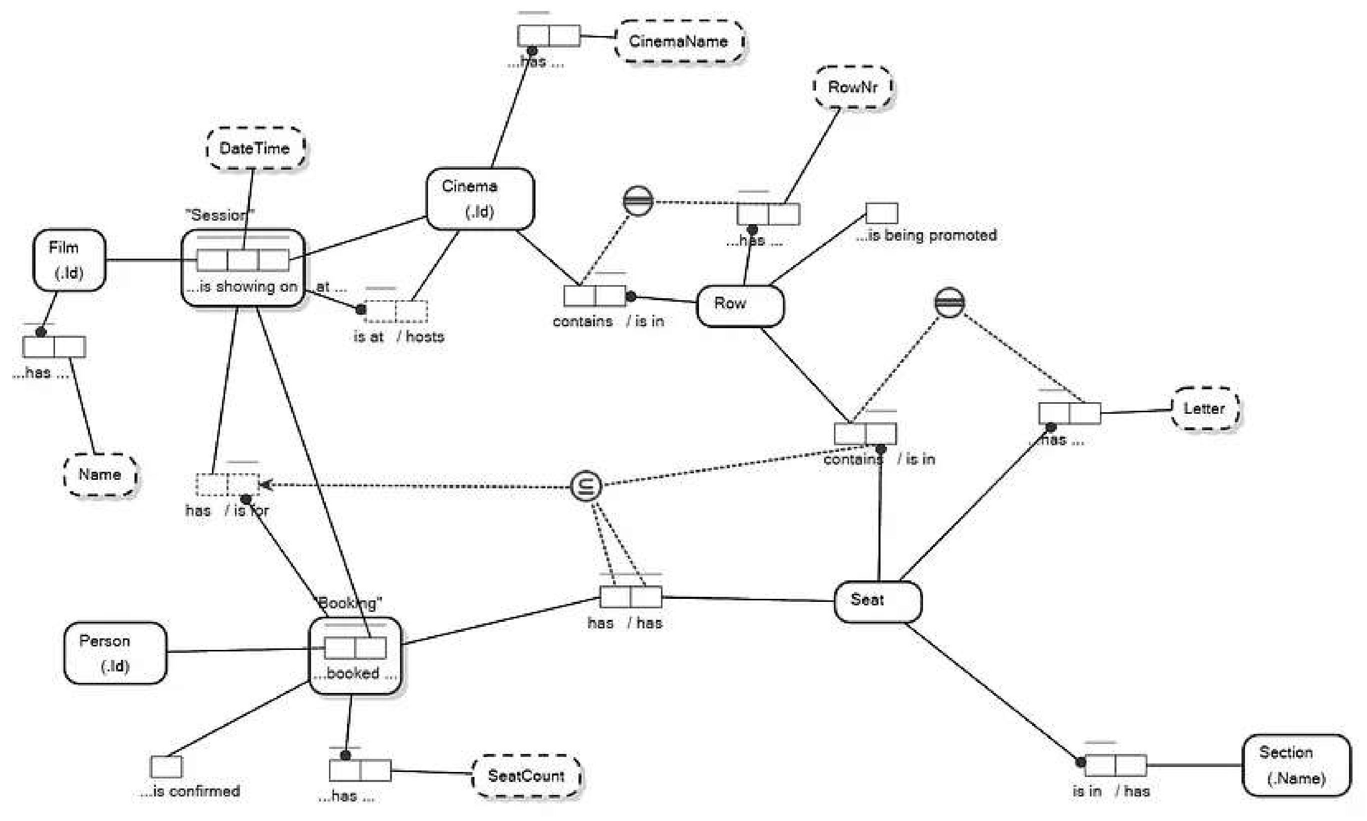
Alternative Abiguous Object-Role Model

If we did not know that our Join Subset Constraint did not follow a path that saw “Cinema contains Row”, but rather “Cinema is promoting Row”, the diagram is ambiguous without the help of software disambiguating the join path. In the diagram above, of course, the software helps disambiguate the diagram by colouring in the join paths. Without the coloured paths, the diagram is ambiguous, as below:



Ambiguous Object-Role Model

Now let us say that we decide to deliberately disambiguate the diagram by moving removing the "Cinema is promoting Row" Fact Type, and simply putting a Unary Fact Type "Row is being promoted" against the Row Entity Type (as below); this does not discount that the formalisation of Object-Role Modeling v2 allows for ambiguous ORM diagrams in the first instance, which is, of course, the central argument of this article.



Disambiguated Object-Role Model

NB I no longer use Terry Halpin's NORMA software for Object-Role Modeling, but I am pretty sure that if you introduce an ambiguous set of join paths for a Join Subset Constraint, the free version of the software will let you know that there is an ambiguity. You might be tempted to say, "Aha, that is sure sign that ORM does not allow ambiguity". No, that merely says the NORMA software is smart enough to let you know that ORM does allow ambiguity and the unwritten implication is that you should upgrade to the professional version of the software which allows you to disambiguate the join path. i.e. There is a fiscal imperative to have you believe you work in the absence of ambiguity, when the logic does not support the claim; at least for ORM2 and Join Subset Constraints.

References

1. Halpin,Terry (1998), UML Data Models from an ORM perspective: Part 7, http://www.orm.net/pdf/ICMArticle1.pdf, Accessed at Monday, 2 May 2022

2. Halpin,Terry (Sept 2005), ORM 2 Graphical Notation, Neumont University, http://www.orm.net/pdf/orm2graphicalnotation.pdf, Accessed at Monday, 2 May 2022

3. Halpin,Terry (Jan 2005), p. 7, Join Constraints, Microsoft Corporation, USA, http://www.orm.net/pdf/joinconstraints.pdf, Accessed at Monday, 2 May 2022

— — — — — — — — — — — — — — — — — —

NB The Cinema Bookings model example used in this article is a derivative work of a model by the same name copyright to Clifford Heath and the ActiveFacts project on GitHub. The original model and license can be found here: https://github.com/cjheath/activefacts-examples/tree/master/cql

Object Role Modeling Formal Languages Symbolic Logic Formalisation Unambiguous