nemo

ontology & conceptual modeling research group

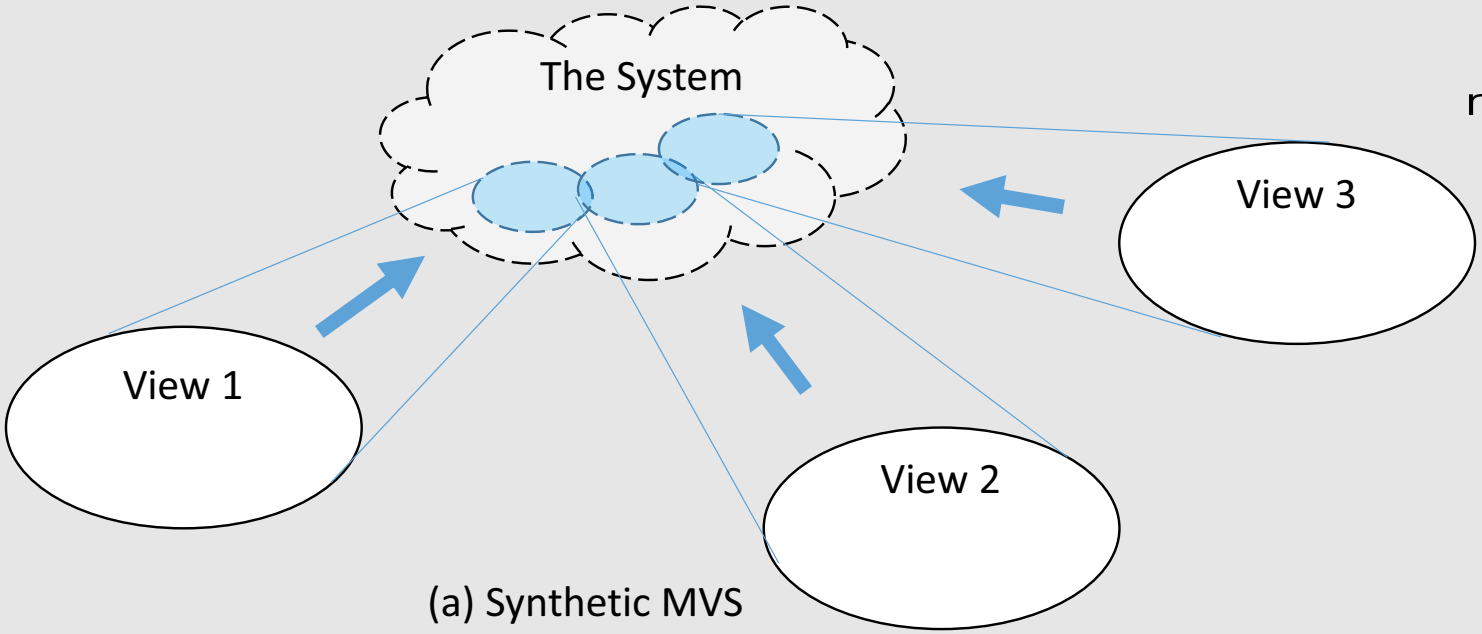
# The role of (reference) ontologies in (view-based) modeling

João Paulo A. Almeida
(jpalmeida@ieee.org)
Federal University of Espírito Santo
http://nemo.inf.ufes.br

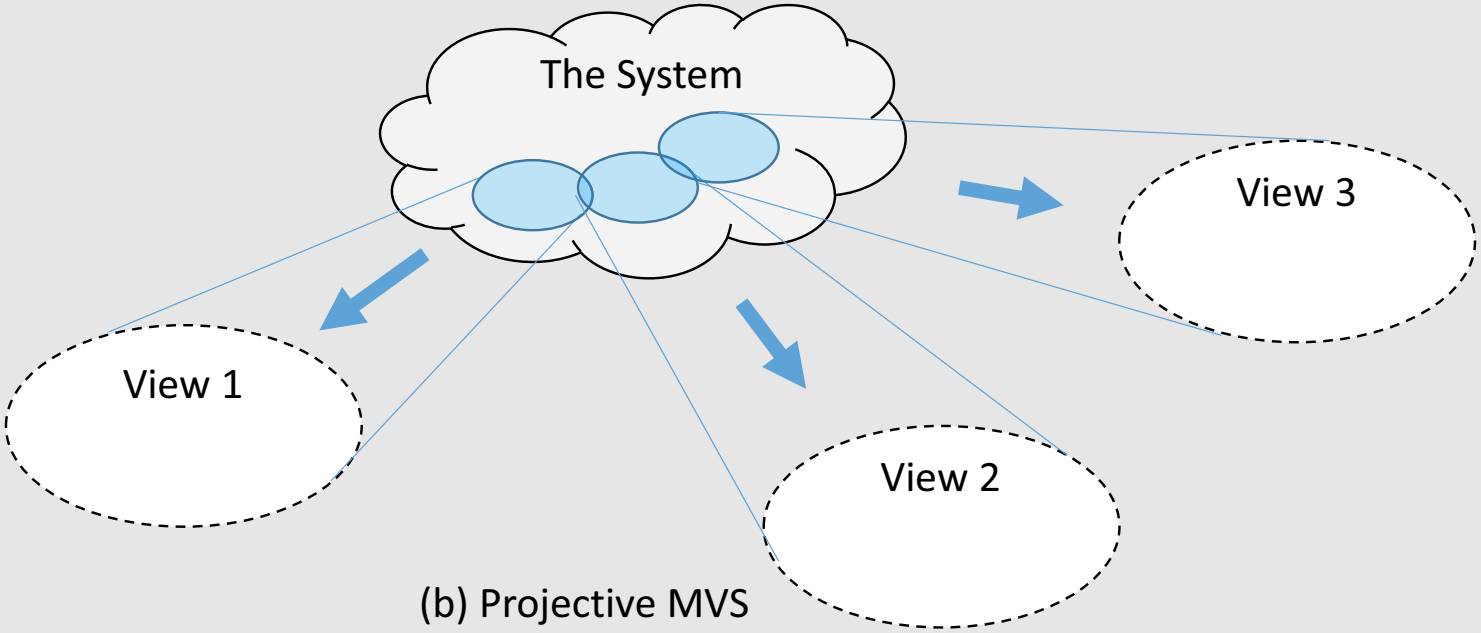
NEMO is currently supported by CNPq, CAPES and FAPES.

# This talk in a nutshell

1. Our models and systems are intended to represent some domain of interest (“domain”)
2. We must then, as best as possible, try to understand this domain
3. This understanding can inform the modeling and system building tasks (and that includes view-related tasks)




(a) Synthetic MVS




(b) Projective MVS

(Atkinson et al, EDOC 2015)

# Reference ontologies

- O-M-G the O word! 😱

- An ontology is a model used to establish admissible states of affairs in that domain (def. inspired in Guarino, 1998)

- Reference ontologies x lightweight ontologies

# Reference ontologies

- Constructed with the **objective of making the best possible description of a certain domain of inquiry**, capturing **a conceptualization** of that domain;
- **Characterize the states-of-affairs which are deemed admissible by a conceptualization;**
- **Not** focused on desirable **computational properties**

# A tiny reference ontology

- Domain of inquiry: person's biological heritage

- In natural language:

① Every person is either a man or a woman
② Every person is the result of a single conception event
③ … that involves the participation of a man and a woman

# A tiny reference ontology – FOL 1/3

- Domain of inquiry: person's biological heritage

- In first-order logics:

① Every person is either a man or a woman

$$\forall x(\text{man}(x) \rightarrow \text{person}(x))$$
$$\forall x(\text{woman}(x) \rightarrow \text{person}(x))$$
$$\forall x(\text{person}(x) \leftrightarrow \text{man}(x) \vee \text{woman}(x))$$
$$\neg \exists x(\text{man}(x) \wedge \text{woman}(x))$$

# A tiny reference ontology – FOL 2/3

- Domain of inquiry: person's biological heritage

- In first-order logics:

② Every person is the result of a single conception event

$$\forall x,y(\text{resultedIn}(x,y)\rightarrow\text{conception}(x)\wedge\text{person}(y))$$

$$\forall x(\text{person}(x)\rightarrow\exists y(\text{conception}(y)\wedge\text{resultedIn}(y,x)))$$
$$\forall x,y,z(\text{person}(x)\wedge\text{resultedIn}(y,x)\wedge\text{resultedIn}(z,x)\rightarrow y=z)$$

$$\forall x(\text{conception}(x)\rightarrow\exists y(\text{person}(y)\wedge\text{resultedIn}(x,y)))$$
$$\forall x,y,z(\text{conception}(x)\wedge\text{resultedIn}(x,y)\wedge\text{resultedIn}(x,z)\rightarrow y=z)$$

# A tiny reference ontology – FOL 3/3

- Domain of inquiry: person's biological heritage

- In first-order logics:

③ … that involves the participation of a man and a woman

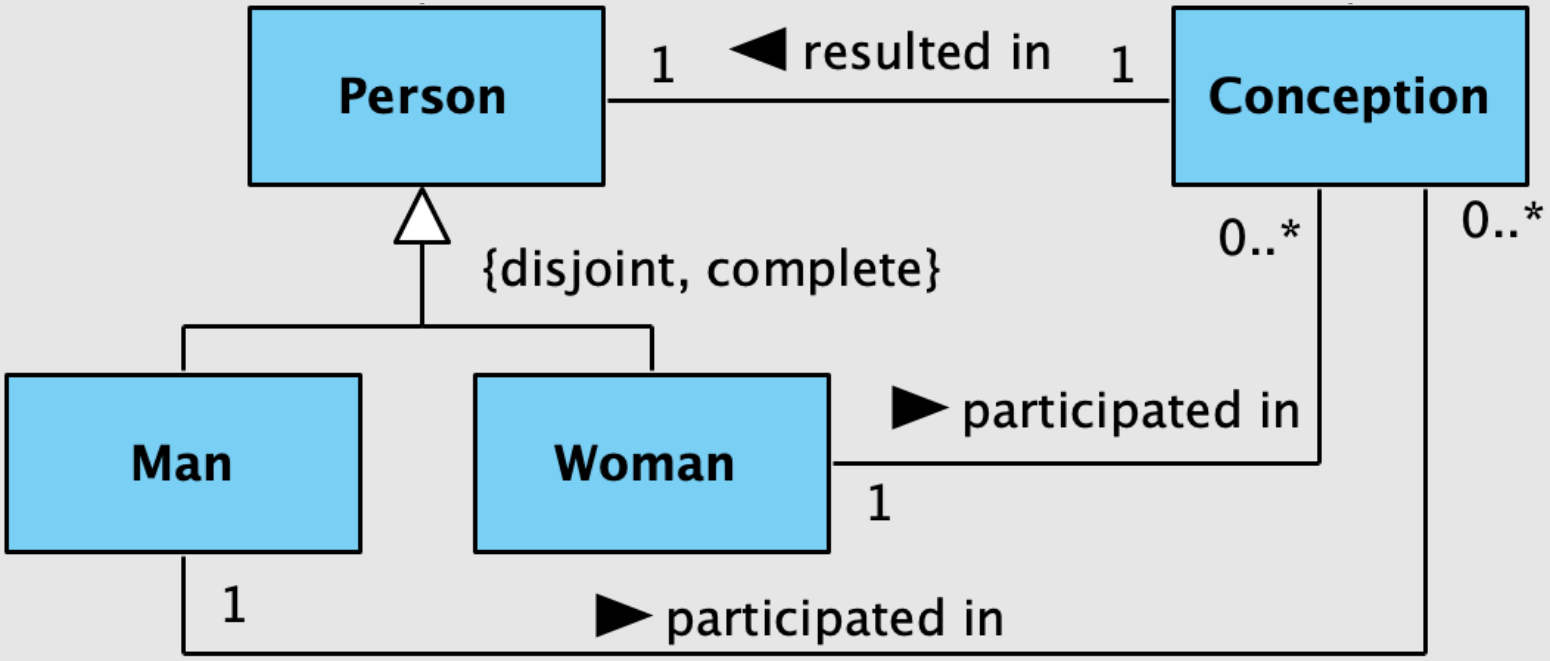
$$\forall x,y(\text{participatedAsFatherIn}(x,y)\rightarrow \text{man}(x)\wedge\text{conception}(y))$$
$$\forall x(\text{conception}(x)\rightarrow\exists y(\text{man}(y)\wedge\text{participatedAsFatherIn}(y,x)))$$
$$\forall x,y,z(\text{conception}(x)\wedge \text{participatedAsFatherIn}(y,x)\wedge\text{participatedAsFatherIn}(z,x)\rightarrow y=z)$$

$$\forall x,y(\text{participatedAsMotherIn}(x,y)\rightarrow \text{woman}(x)\wedge\text{conception}(y))$$
$$\forall x(\text{conception}(x)\rightarrow\exists y(\text{woman}(y)\wedge\text{participatedAsMotherIn}(y,x)))$$
$$\forall x,y,z(\text{conception}(x)\wedge \text{participatedAsMotherIn}(y,x)\wedge\text{participatedAsMotherIn}(z,x)\rightarrow y=z)$$

# A tiny reference ontology – UML

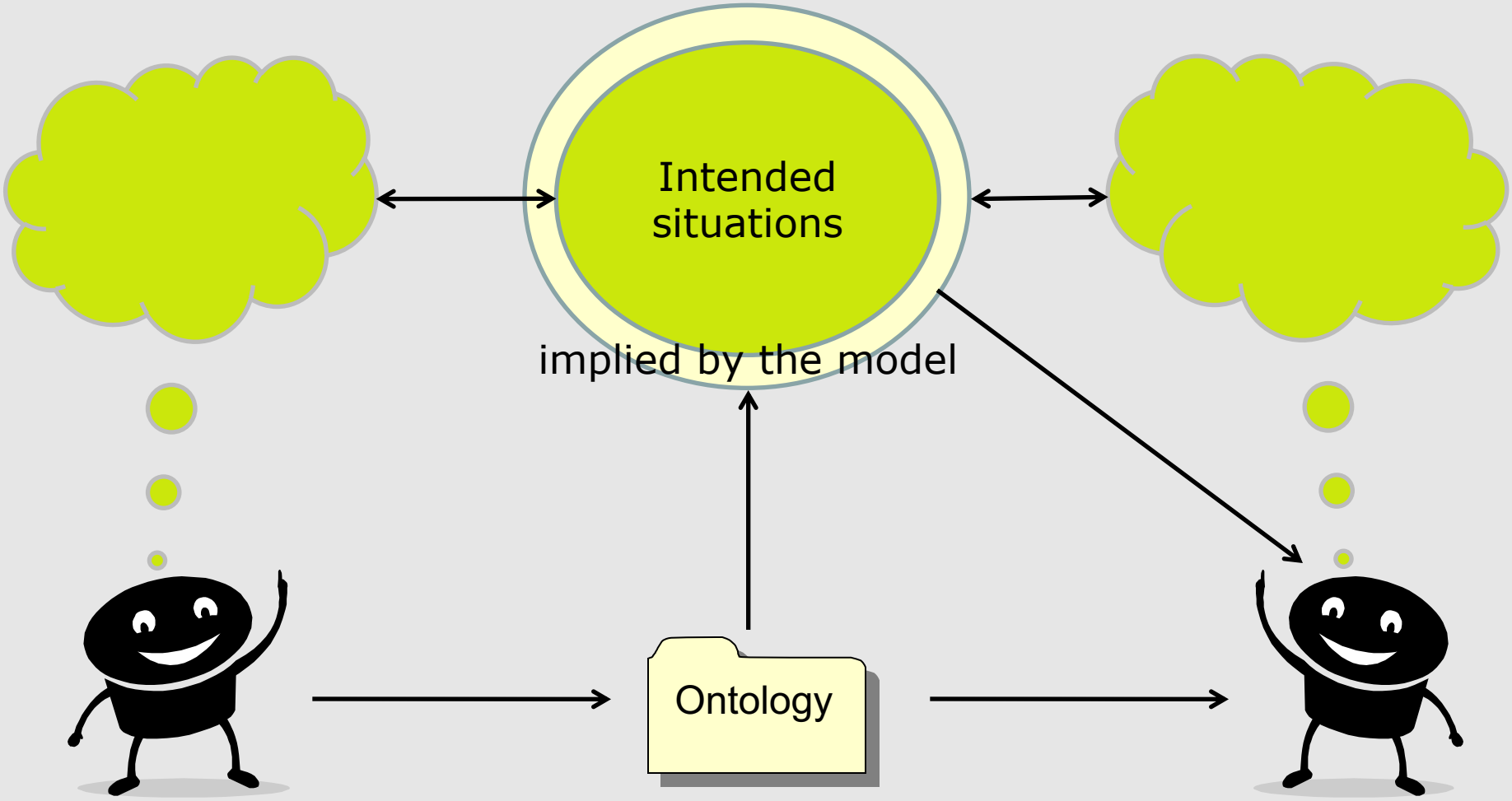
- Domain of inquiry: person's biological heritage

- In UML (assuming a suitable formal semantics):



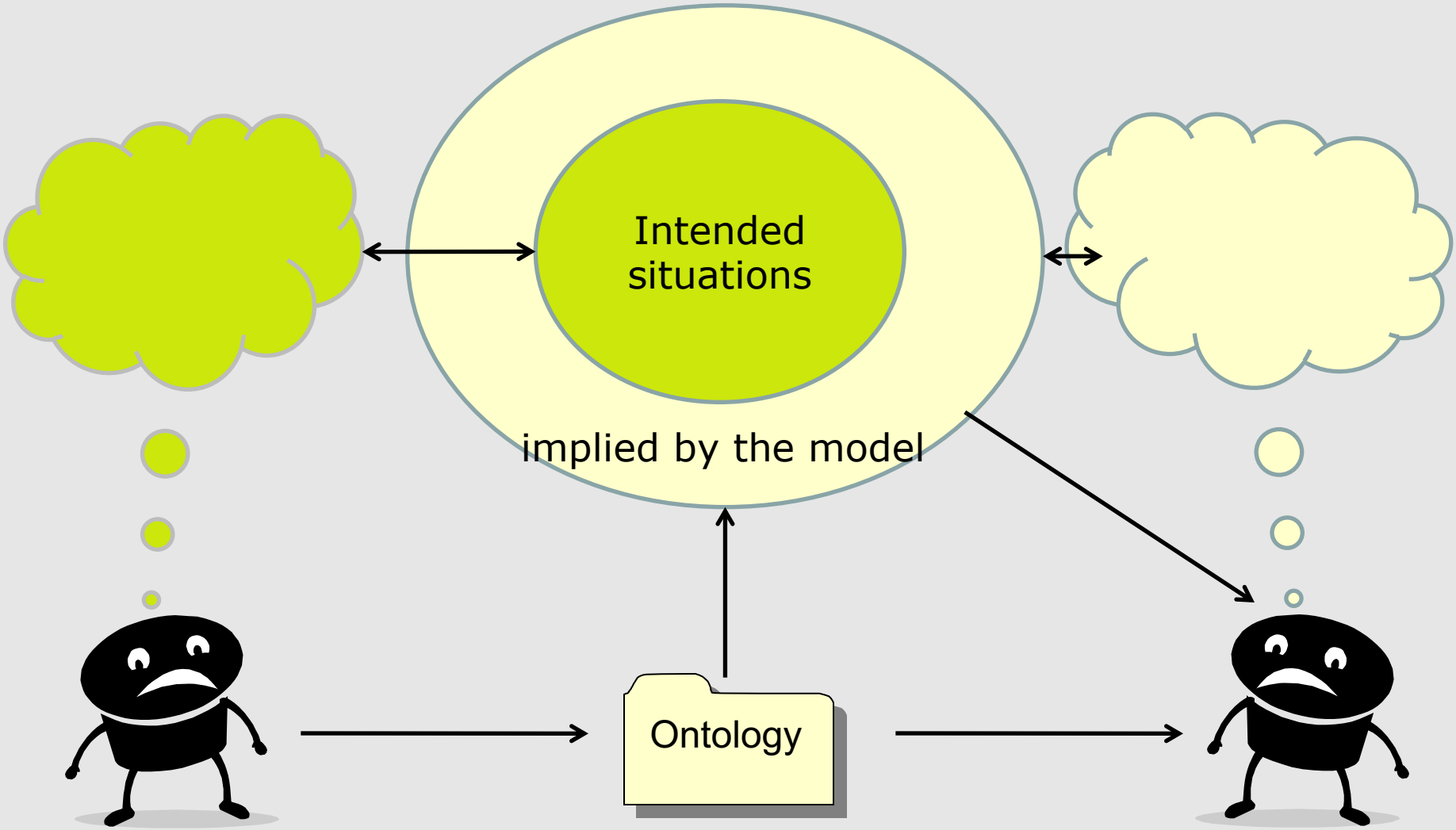
# What makes a good reference ontology?

- Precision



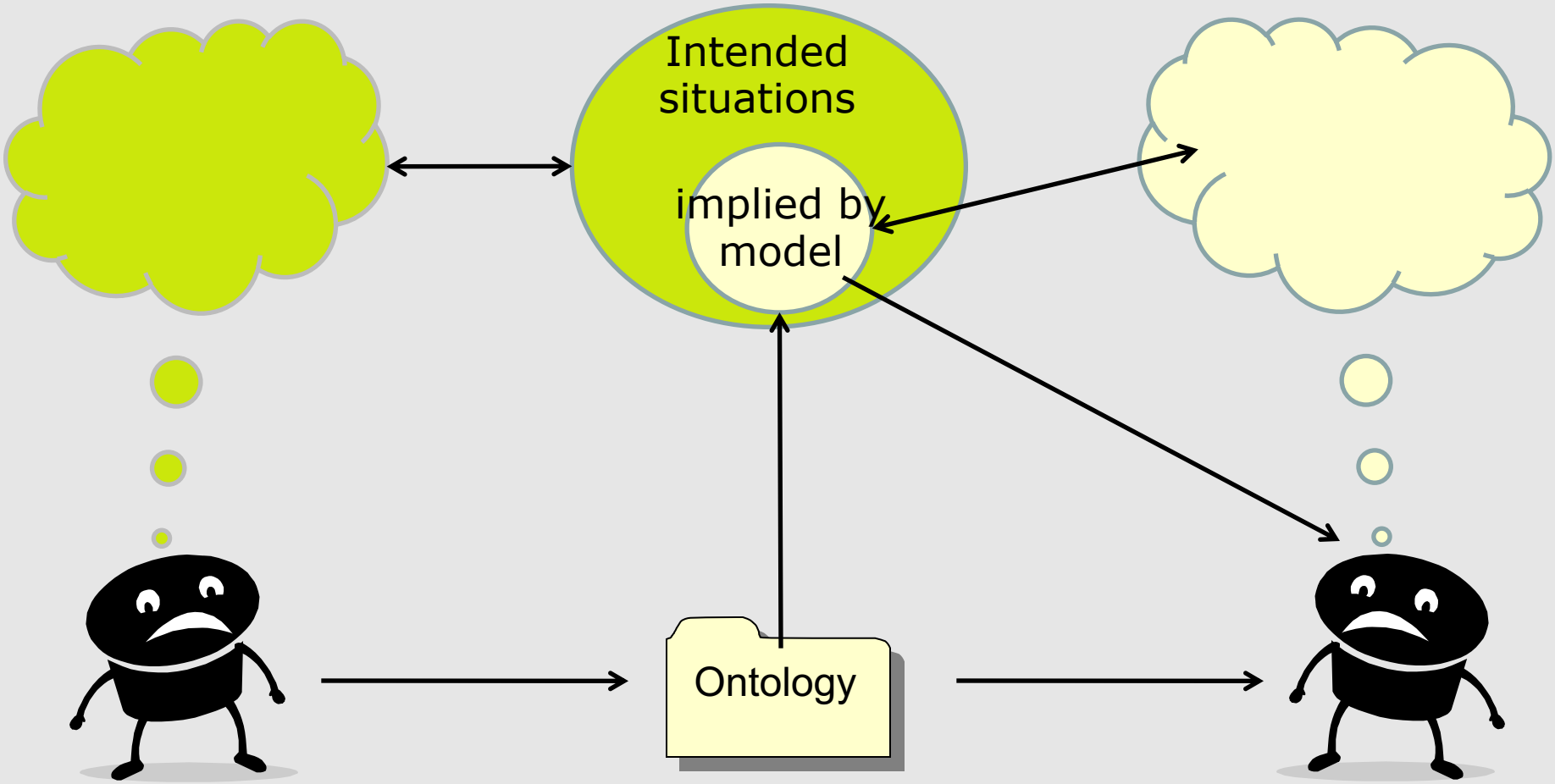
# What makes a **bad** reference ontology?

- Underconstraining! Ontology too permissive…



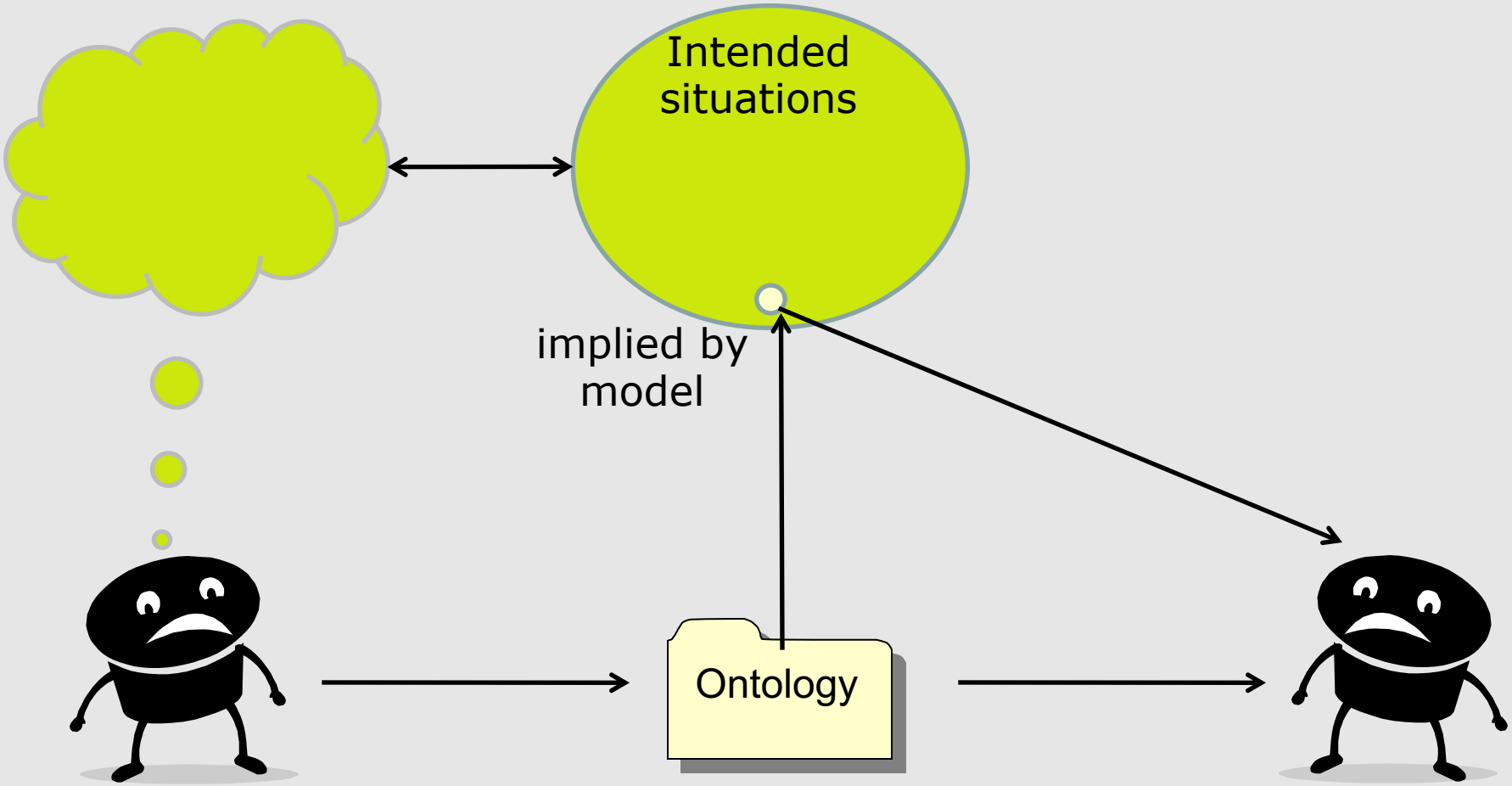
# What makes a **bad** reference ontology?

- Overconstraining! Ontology rules out admissible situations…



# What makes a terrible reference ontology?

- Inconsistency…



# We need all the help we can get

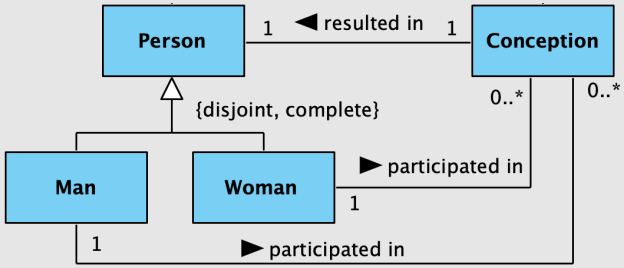
- Verification
  - Check consistency (can we find situations that satisfy all the axioms?)
  - Check assertions

- Validation (with respect to intented conceptualization)
  - Check underconstraining (do we rule out what we intend to)
  - Check overconstraining (do we admit what we intend to)

- “Model finding” (exemplified with Alloy):
  1. Ontology simulation
  2. Projective visualization

3. Reuse of foundational ontology

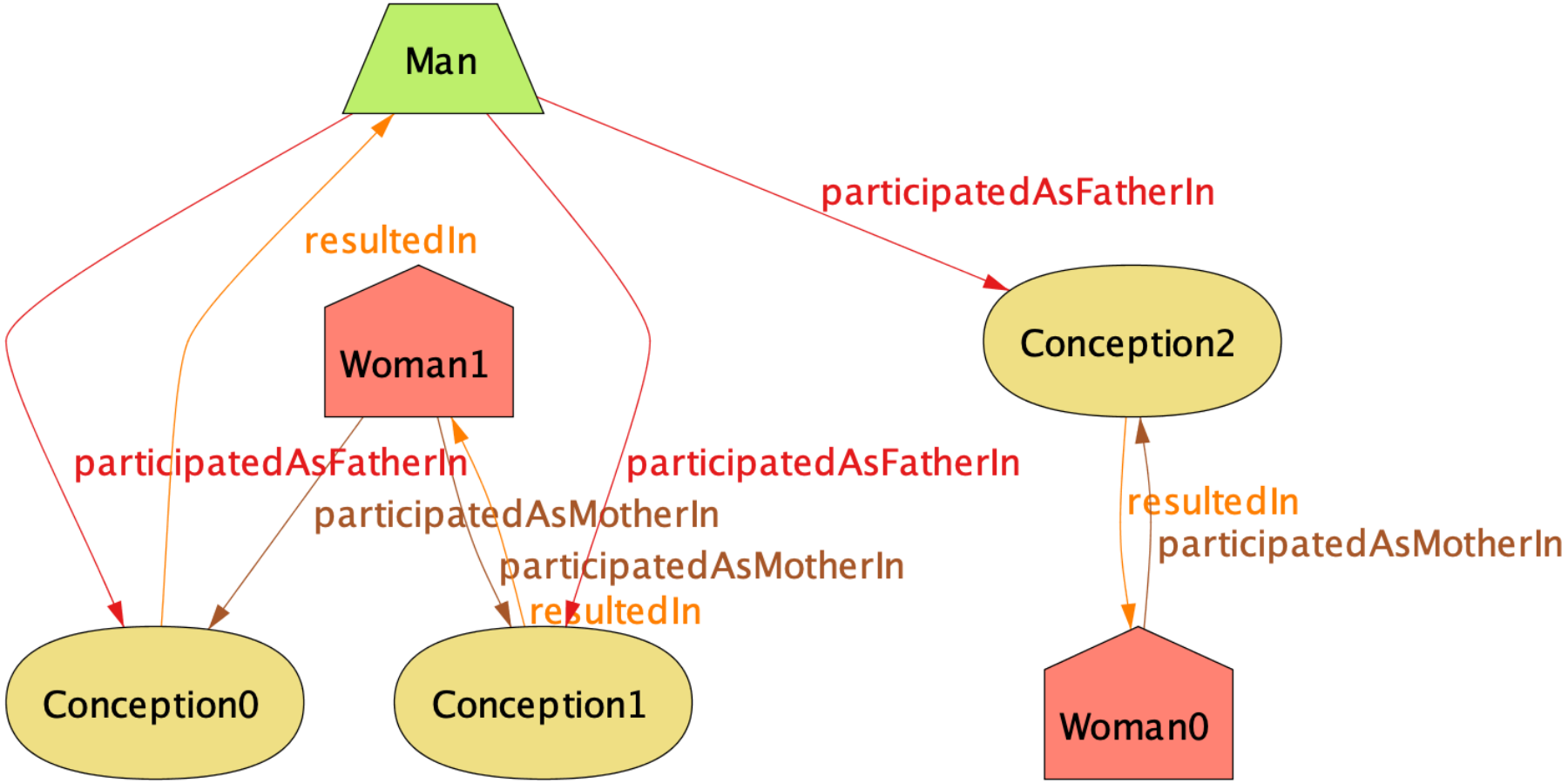
# Simulate our ontology




```
abstract sig Person { }
sig Man extends Person {
    participatedAsFatherIn : set Conception
}
sig Woman extends Person {
    participatedAsMotherIn : set Conception
}
sig Conception {
    resultedIn : one Person
}
fact {
    all p : Person | one resultedIn.p and
    all c : Conception |
        one participatedAsMotherIn.c and
        one participatedAsFatherIn.c
}

run { }
```

# Simulate our ontology

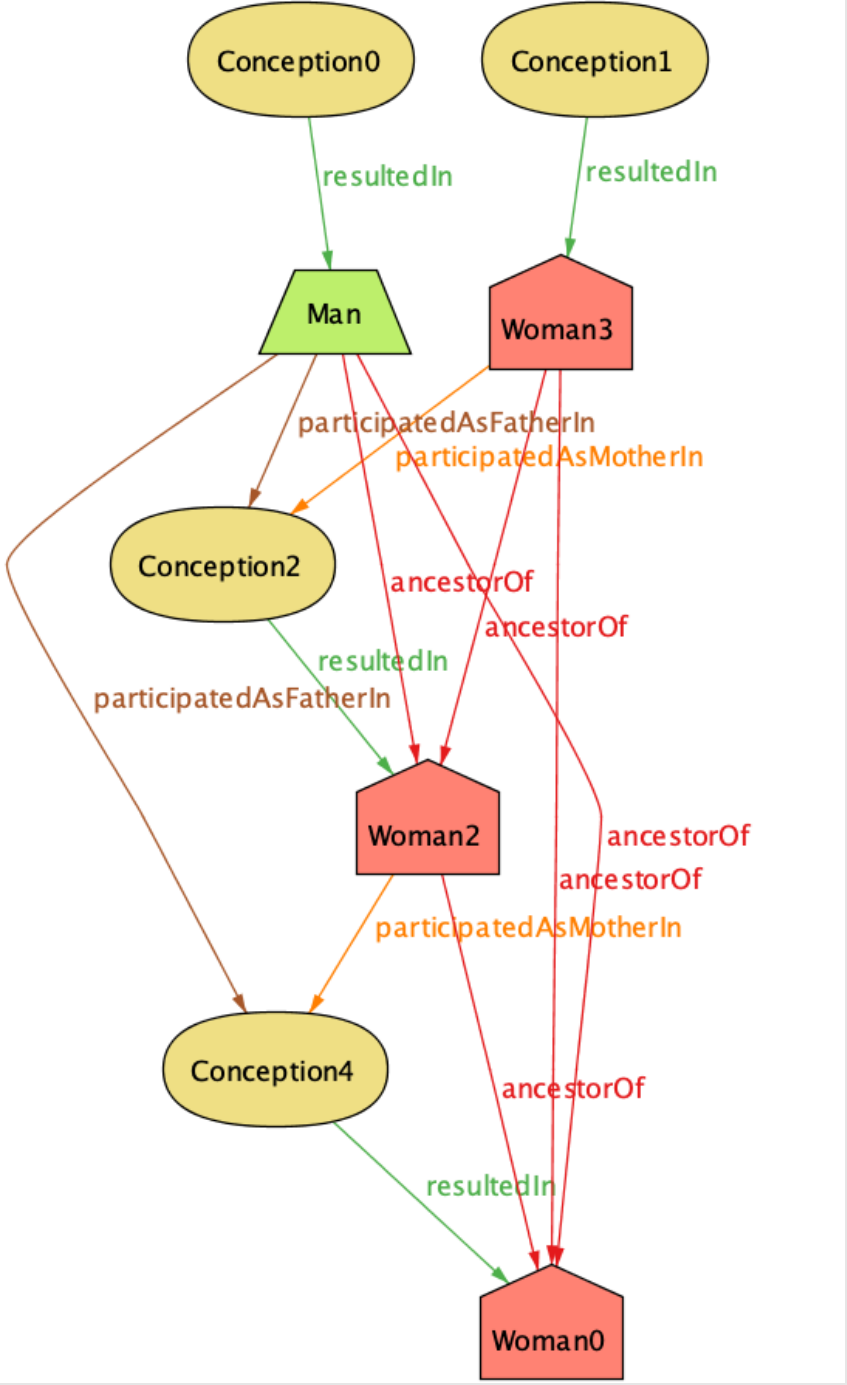



underconstrained…

# Fix our ontology to improve precision

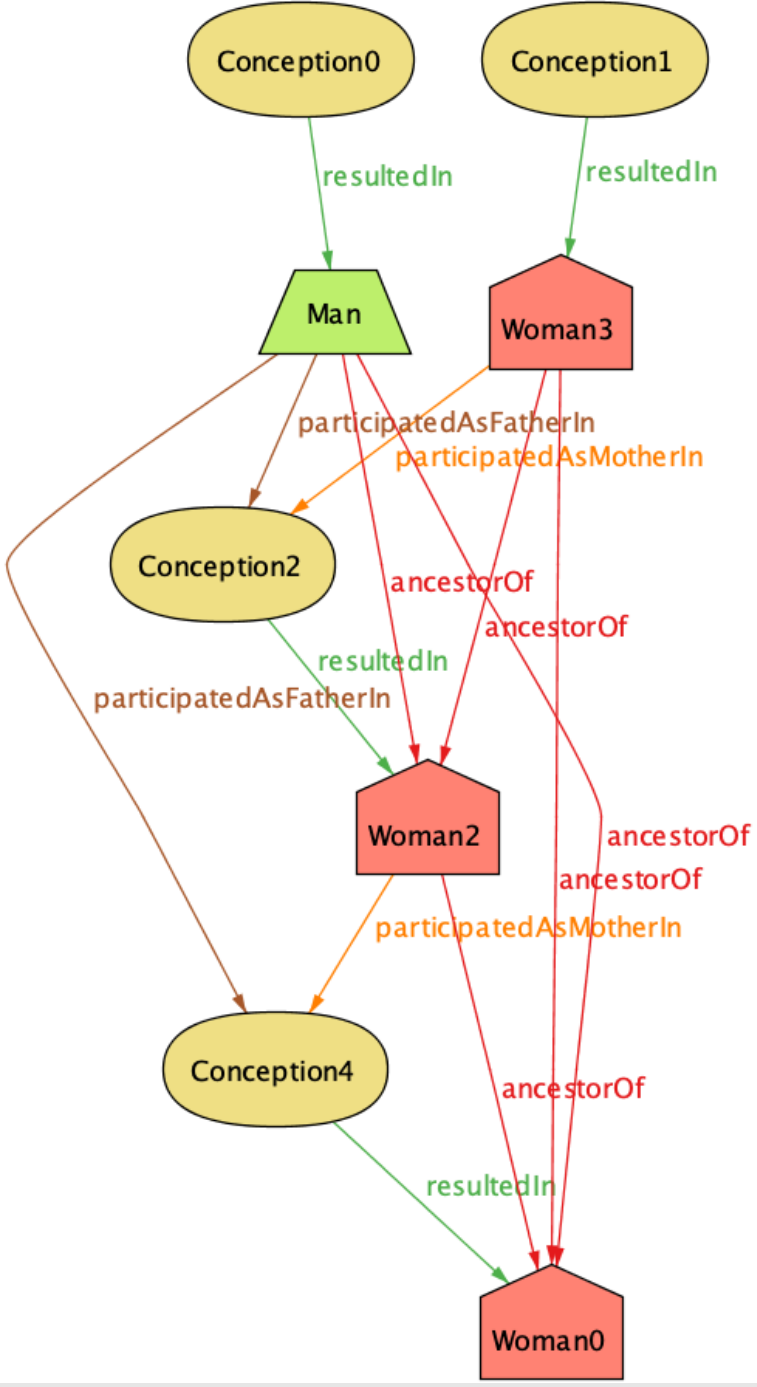
- Definition of fatherOf, motherOf, parentOf, ancestorOf
  - "Derivations"
  - $\forall x,y(\text{fatherOf}(x,y) \leftrightarrow \exists z(\text{conception}(z) \wedge \text{participatedAsFatherIn}(x,z) \wedge \text{resultedIn}(z,y)))$
  - $\forall x,y(\text{motherOf}(x,y) \leftrightarrow \exists z(\text{conception}(z) \wedge \text{participatedAsMotherIn}(x,z) \wedge \text{resultedIn}(z,y)))$
  - $\forall x,y(\text{parentOf}(x,y) \leftrightarrow (\text{motherOf}(x,y) \vee \text{fatherOf}(x,y)))$
  - ancestorOf transitive closure of parentOf

- Axioms to rule out inadmissible states
  - parentOf acyclic (and as theorems ancestorOf acyclic, fatherOf, motherOf, parentOf irreflexive)

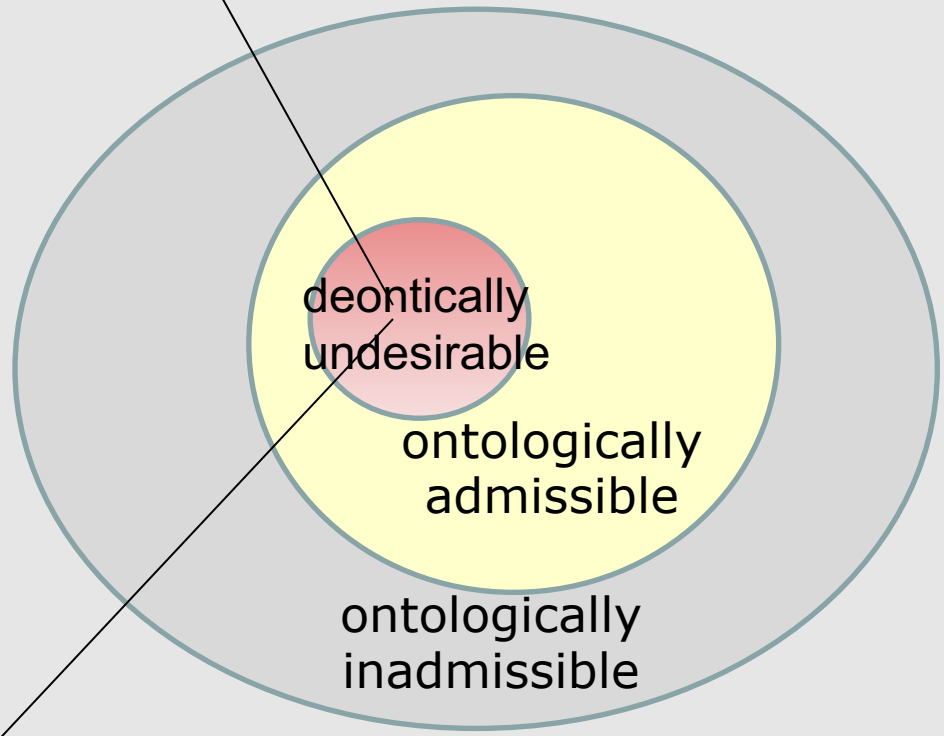



nemo

parentOf, fatherOf, motherOf hidden




ontologically inadmissible situations
≠
deontically undesirable situations



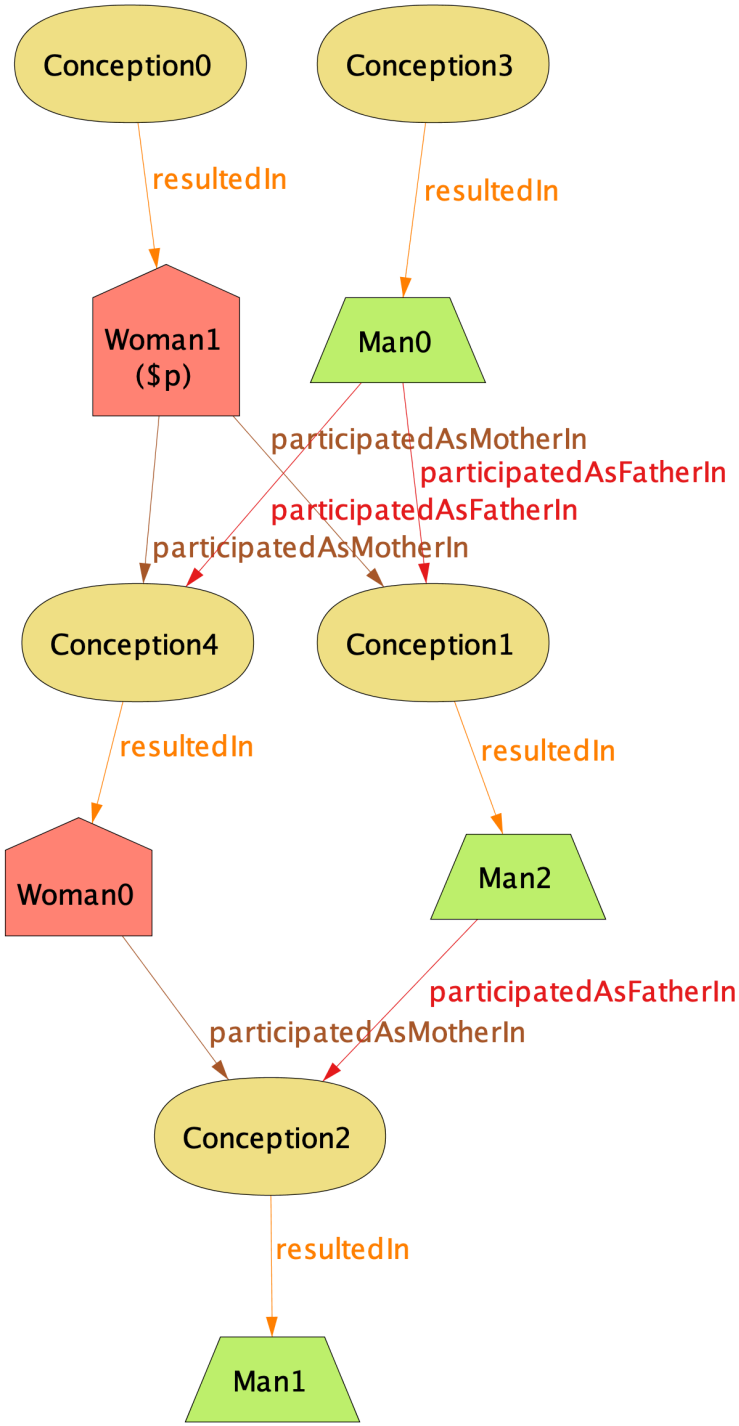

parentOf, fatherOf, motherOf hidden

# We need all the help we can get (2/3)

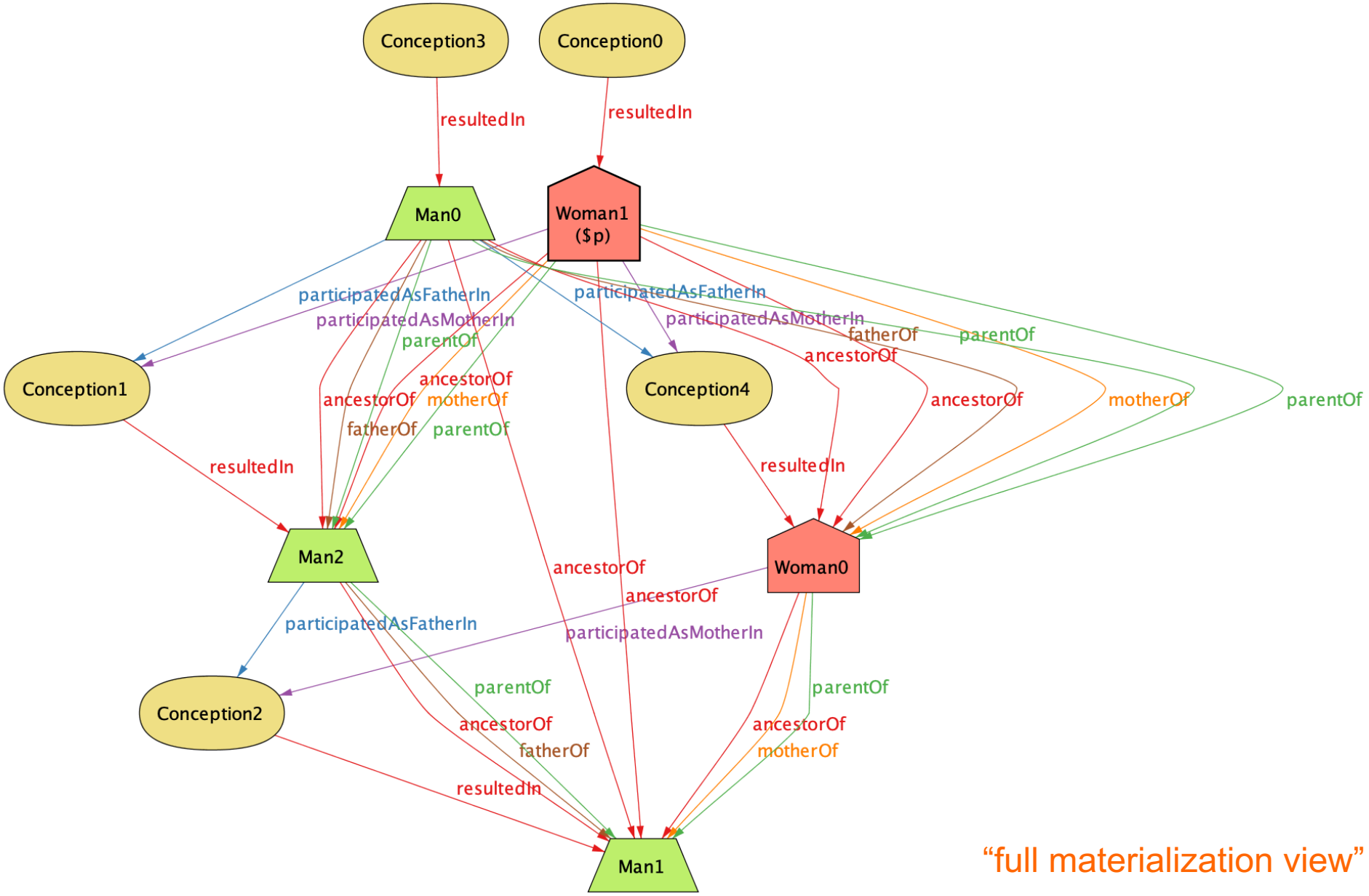
- Including projective view-based strategies
- (Not yet talking about designing view-based environments, but using views in the design of the ontology)

“truthmaker view”

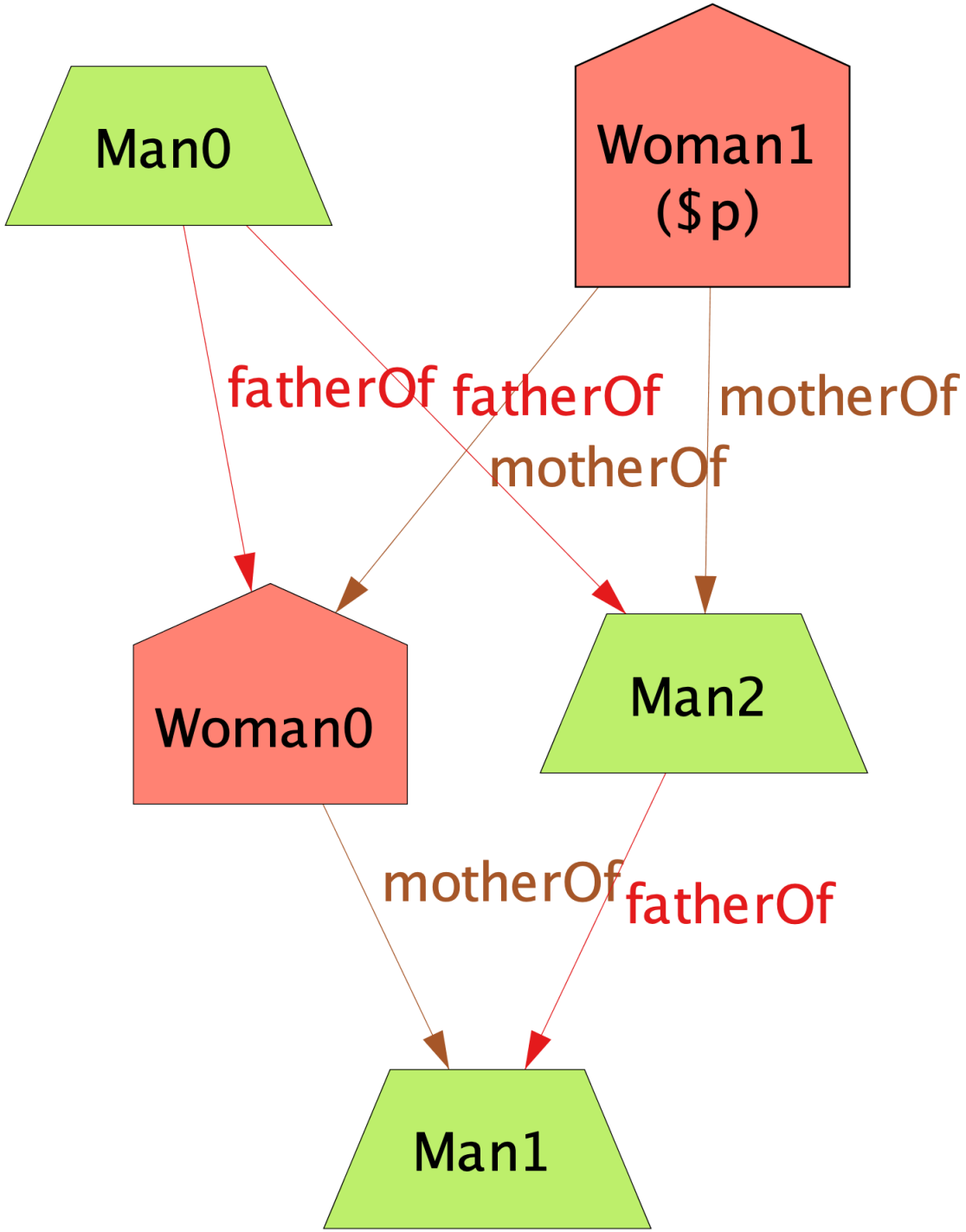
(left out all that can be entailed: fatherOf, parentOf, motherOf)

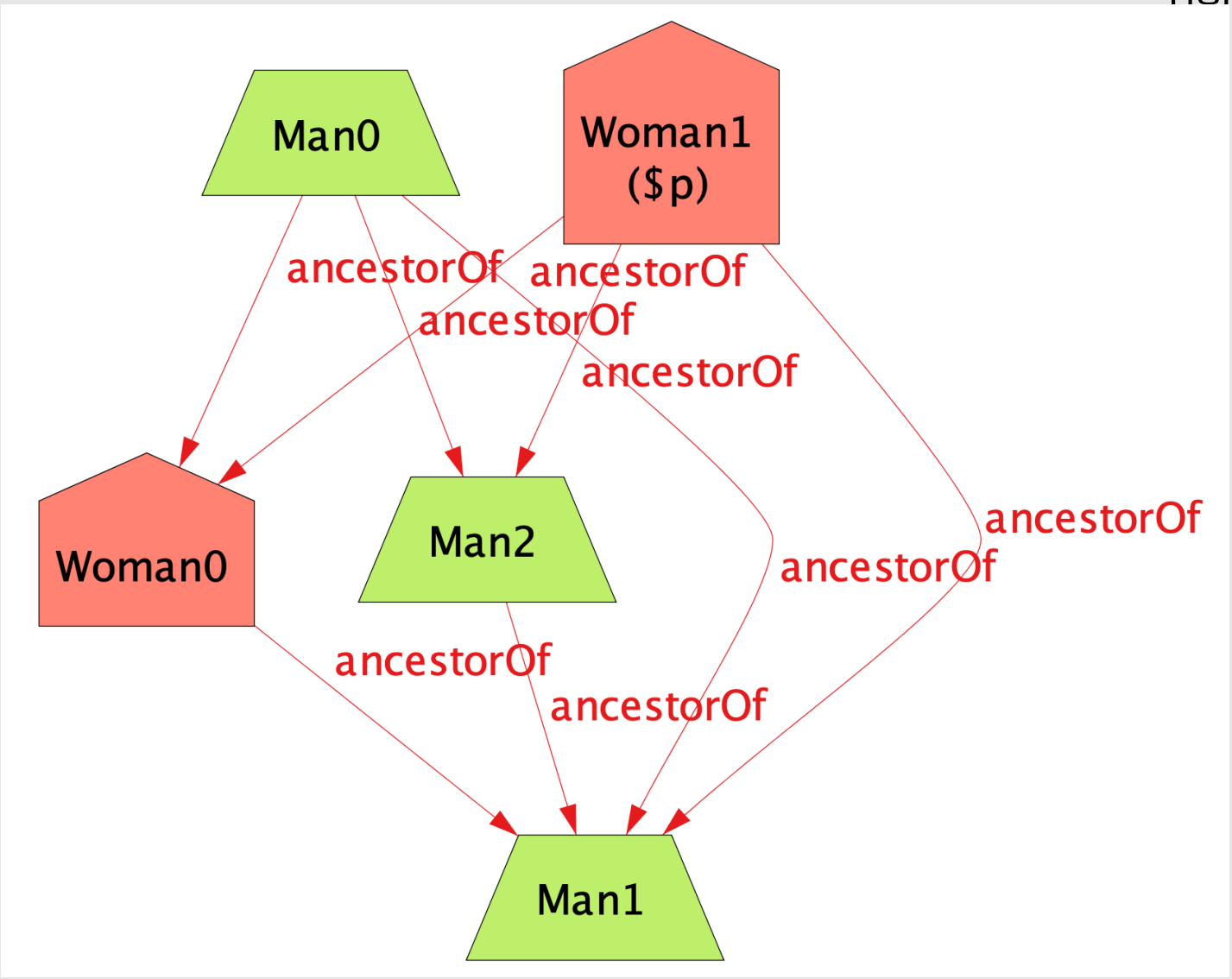



again…
deontically undesirable



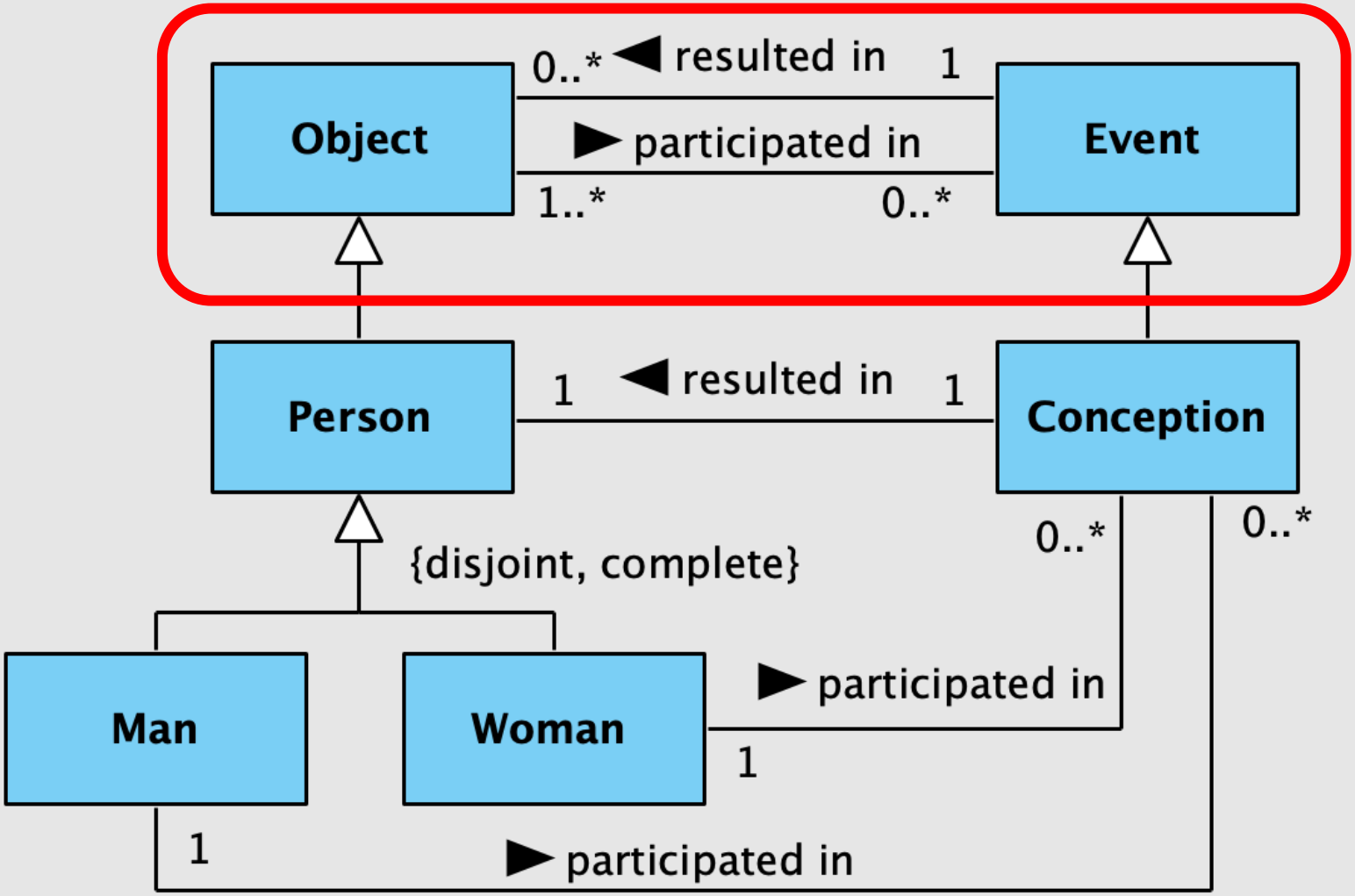

"full materialization view"


Man0
Woman1
($p)
fatherOf
fatherOf
motherOf
motherOf
Woman0
Man2
motherOf
fatherOf
Man1


Man0
Woman1
($p)
ancestorOf
ancestorOf
ancestorOf
ancestorOf
Woman0
Man2
ancestorOf
ancestorOf
ancestorOf
ancestorOf
Man1

# We need all the help we can get (3/3)

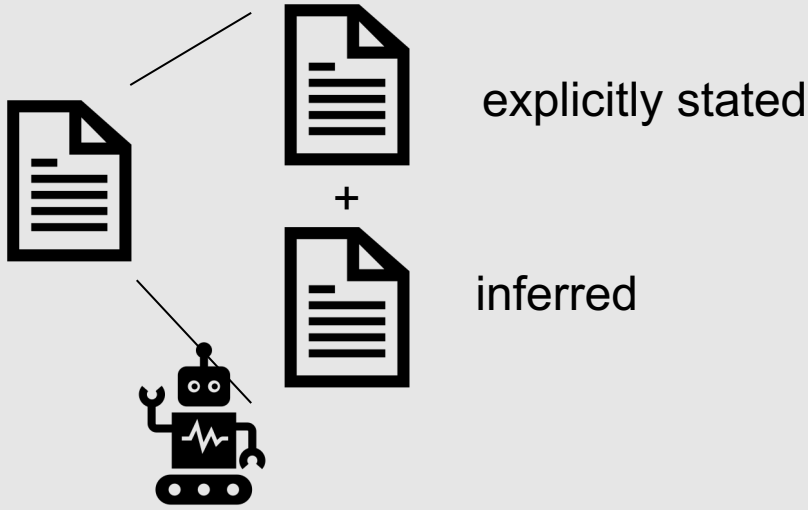
domain-independent notions



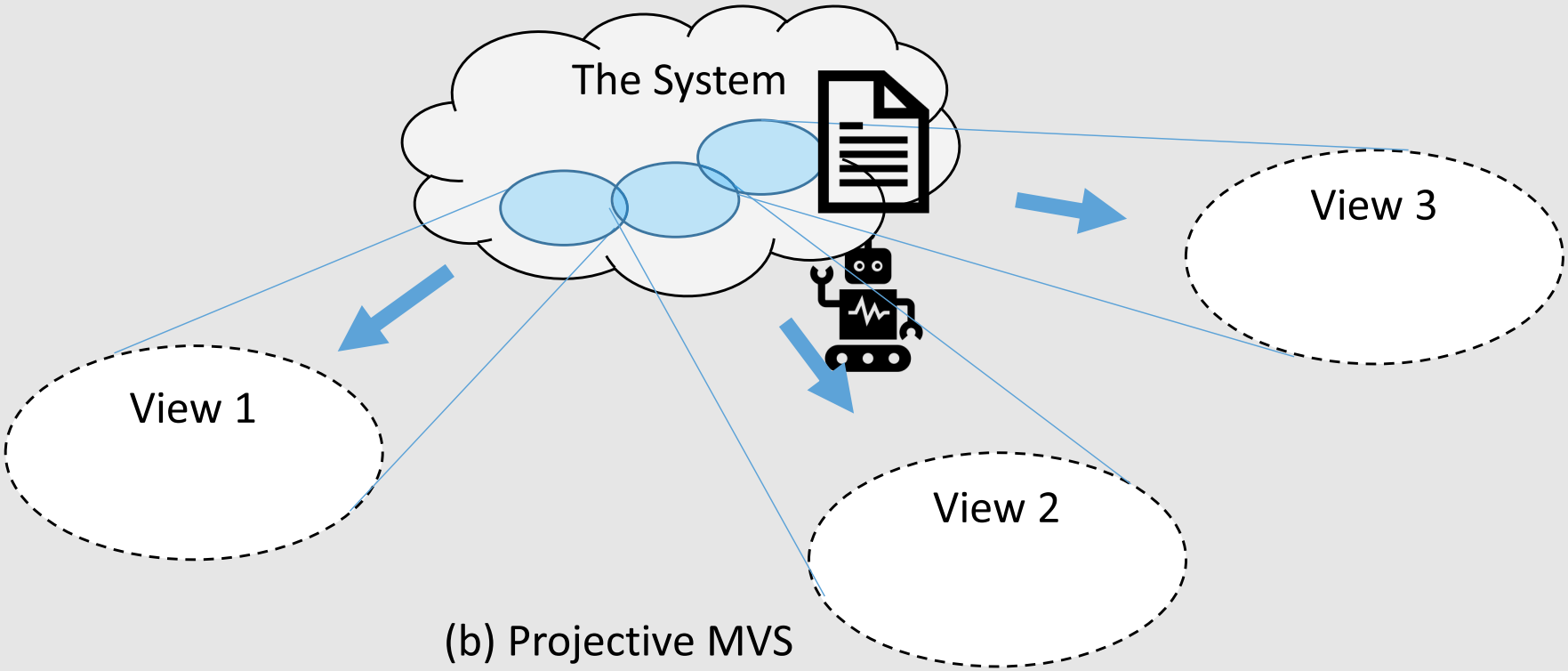

ancestorOf is now acyclic as a consequence of events forming a partial order + Objects have to be created first to participate in an event later

# Ontology in a knowledge base

- Statements articulated using the ontology (A-box):
  - motherOf(Poliana, João Paulo)
  - fatherOf(João Paulo, Eduardo)
  - Therefore, we can infer or entail (virtual A-box):
    - man(João Paulo), person(João Paulo)
    - woman(Poliana), person(Poliana)
    - person(Eduardo)
    - ancestorOf(Poliana, Eduardo)



# Ontologies and projective views

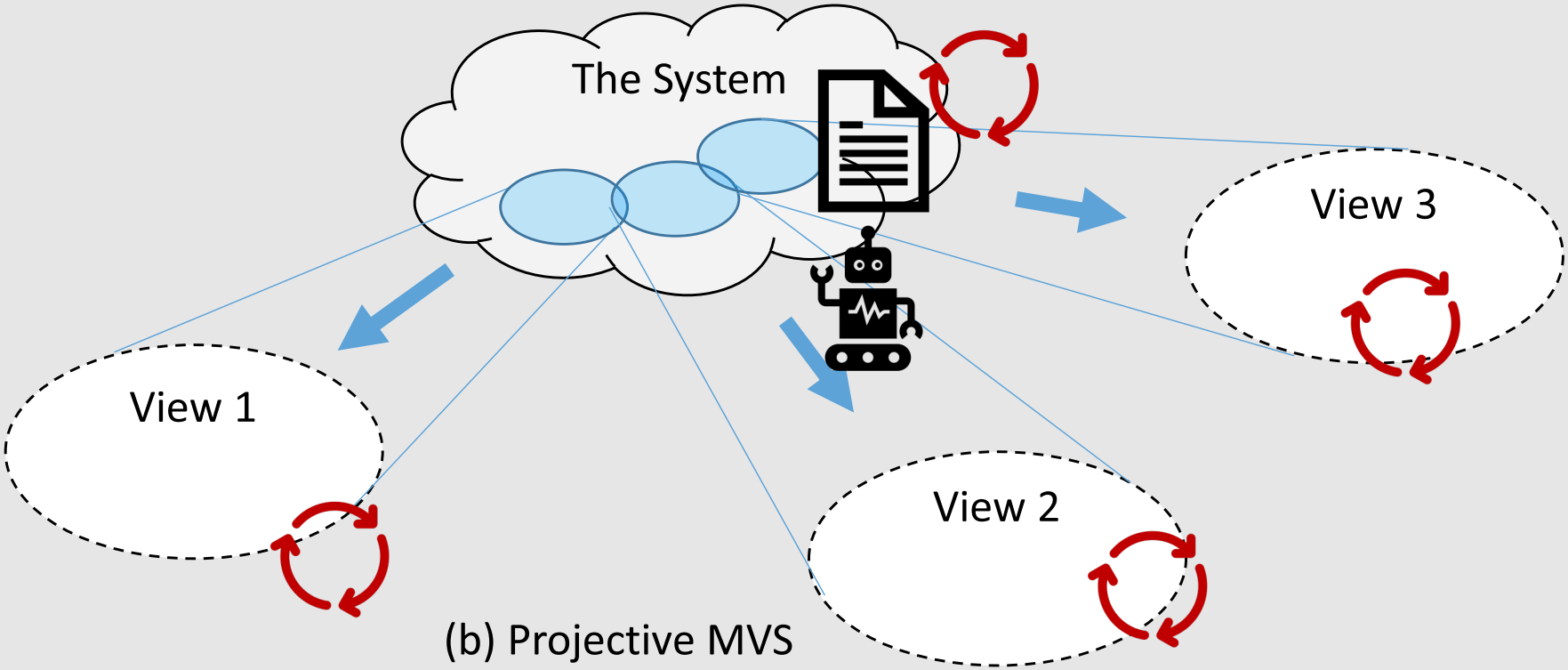



(b) Projective MVS

(Atkinson et al, EDOC 2015)

# Ontologies and projective views

- An Ontology-based SUM (a semantic SUM)




(b) Projective MVS

Constraint Maintenance and View Updating Techniques in deductive databases

(Atkinson et al, EDOC 2015)

# Lightweight ontologies

- Typically specified in languages with limited expressiveness
- “Reasoning” (logical inferencing, entailment) becomes tractable

- RDFS
- OWL 2 – DL (SROIQ)
- Datalog (deductive databases)

① Sacrifice precision for computational properties

② Even with the limited expressiveness, a general solution for update maintenance under entailment is not there

# ① Sacrificing Precision for Computation

- Choice of constructs
  - Subsumption (set containment)
  - Complement, union, intersection
  - Domain and range for binary relations
  - Relations can be declared, symmetric/assymmetric, transitive, reflexive/irreflexive, pairwise disjoint
  - Composition and subsumption for relations
- Syntactic constraints to ensure decidability: regularity and simplicity

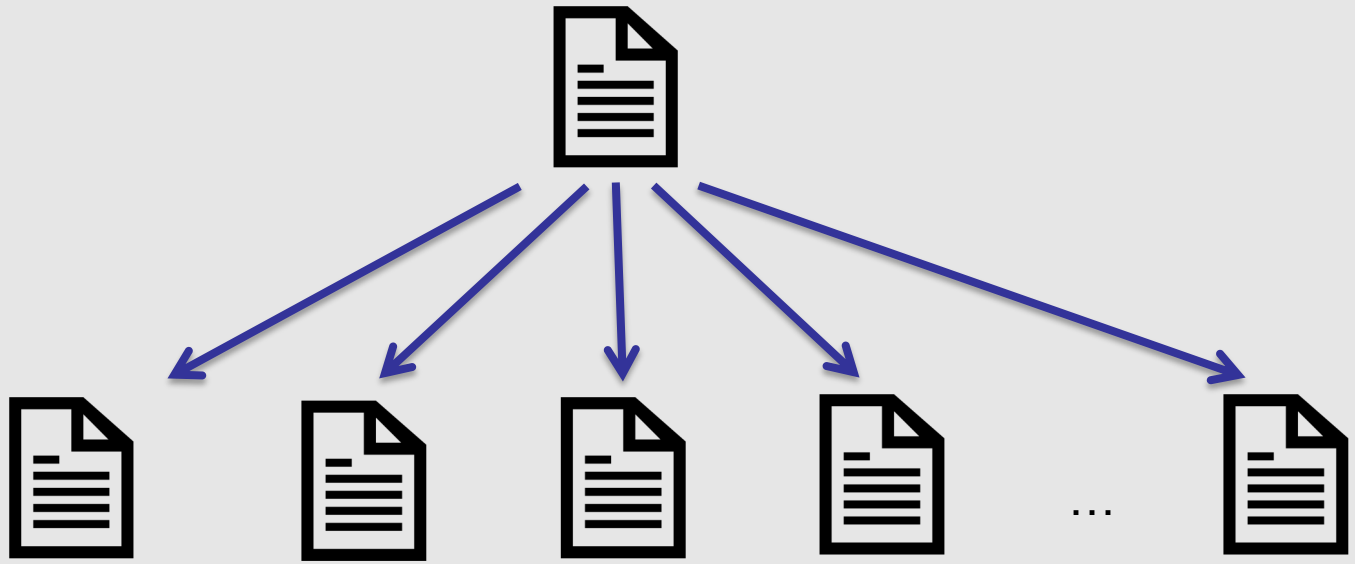
# Precision in OWL-DL SROIQ is hard

- Specified a reference ontology of events in FOL (UFO-B)
- Transformed into 185 axioms using constructs of OWL-DL
  - Already loosing expressiveness in this process
- Checked all combinations which maximize precision while respecting SROIQ constraints
  - Defined a methodology for that
  - 12,288 possible combinations!!!
  - What do you want to lose from the ontology? Pick your poison!

https://content.iospress.com/articles/applied-ontology/ao190214

# Which poison to pick? Just 2 examples…

- allow the existence of an Event whose endPoint precedes the beginPoint of another, but does not happen before the second Event
- allow a ComplexEvent to have an Event as part without mereologically overlapping it




12,288 of those

# ② Updates and entailment

- What does it mean if an implied fact is explicitly (re)inserted (or deleted)?
- Which (if any) additional facts should be inserted (or resp. deleted) upon updates?
- How to treat inconsistencies arising through updates?

- Two main techniques:
  - Materialize inferences (“materialized RDF stores”)
    - think non-essential SUM
  - Compute answers at query runtime (“reduced RDF stores”)
    - keep SUM essential, corresponding to “truthmaker view”
    - compute at view projection time, possibly rewriting queries

Ahmeti, Calvanese, Polleres, ISWC 2014
https://link.springer.com/chapter/10.1007/978-3-319-11964-9_28

# Updating RDFS ABoxes and TBoxes in SPARQL

Albin Ahmeti[1], Diego Calvanese[2], and Axel Polleres[3]

For the sake of this paper, we address such questions with the focus on a deliberately minimal ontology language, namely the minimal RDFS fragment of [19].[4] As it turns out,

[4] We ignore issues like axiomatic triples [13], blank nodes [17], or, in the context of OWL, inconsistencies arising through updates [5]. Neither do we consider named graphs in SPARQL, which is why we talk about "triple stores" as opposed to "graph stores" [8].

even in this confined setting, updates as defined in the SPARQL 1.1 Update specification impose non-trivial challenges; in particular, specific issues arise through the interplay of INSERT, DELETE, and WHERE clauses within a single SPARQL update operation, which

## 9 Conclusions

We have presented possible semantics of SPARQL 1.1 Update in the context of RDFS. To the best of our knowledge, this is the first work to discuss how to combine RDFS with the new SPARQL 1.1 Update language. While we have been operating on a very restricted setting (only capturing minimal RDFS entailments, restricting BGPs to disallow non-standard use of the RDFS vocabulary), we could demonstrate that even in this setting the definition of a SPARQL 1.1 Update semantics under entailments is a non-trivial task. We proposed several possible semantics, neither of which might seem intuitive for all possible use cases; this suggests that there is no "one-size-fits-all" update semantics.

ISWC 2014 https://link.springer.com/chapter/10.1007/978-3-319-11964-9_28

# One size does not fit all

- Design “syntactic” SUM informed by reference ontology

- Metamodel design informed by reference ontology as a semantic background
  - Semantically-motivated syntactic constraints for a language arise from axioms
  - Semantically-motivated views arise from axioms
  - Model updates informed by reference ontology

- What is the difference between an ontology and a metamodel?
  - Confusion arises from similar languages used for their definition

# Two kinds of tasks

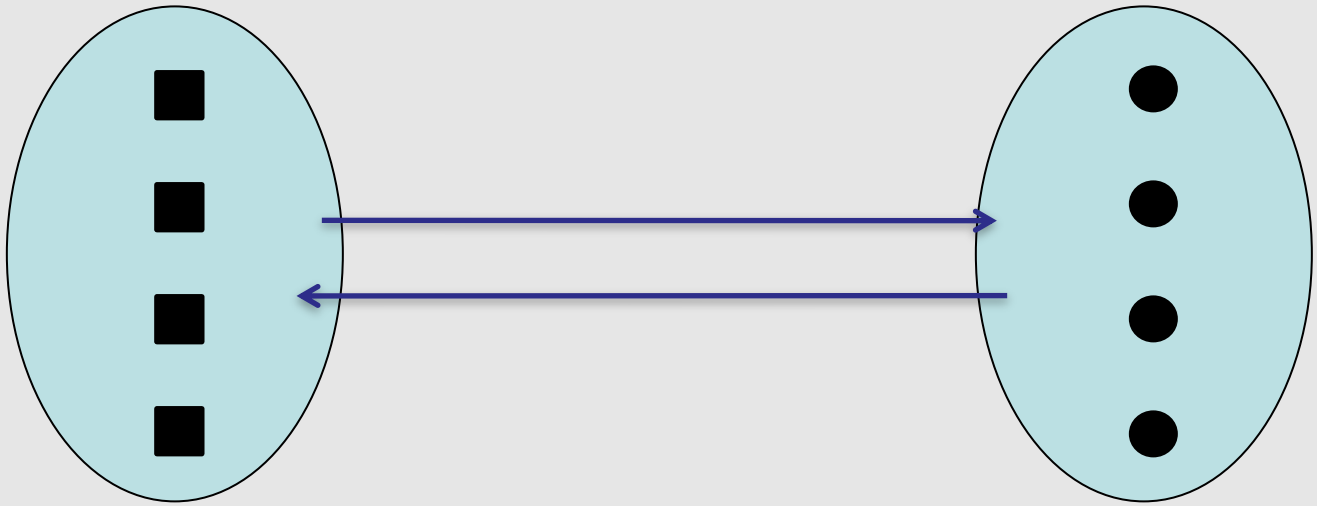
- Language and Tool Engineering
  - Models are used to define abstract syntax of languages
- Ontology Engineering (some people call this Conceptual Modeling)
  - Models used to capture meaning postulates, specify domain conceptualization
  - May serve to establish articulate the semantics of a language

- Different purposes (complementary!)
- But similar (same?) object-oriented representation scheme
  - Types and instances
  - Classes and objects
  - Universals and particulars

# Syntax, Semantics and All that Stuff[1]

Theory of the (abstract) syntax

Theory of the subject domain



Concerned with syntactically valid models

Explicitly represented in a language metamodel / grammar (mixed)

Concerned with admissible/possible entities in a subject domain

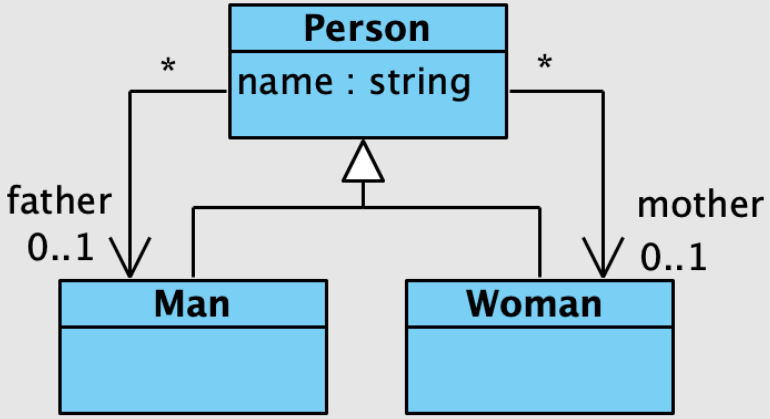
Explicitly represented in a reference ontology (or ontology-based conceptual model)

1 This is a reference to the title of a paper by David Harel and Bernhard Rumpe: Modeling Languages: Syntax, Semantics and All That Stuff, 2000.

# Syntax, Semantics and All that Stuff

Theory of the syntax

Theory of the subject domain




Language metamodel

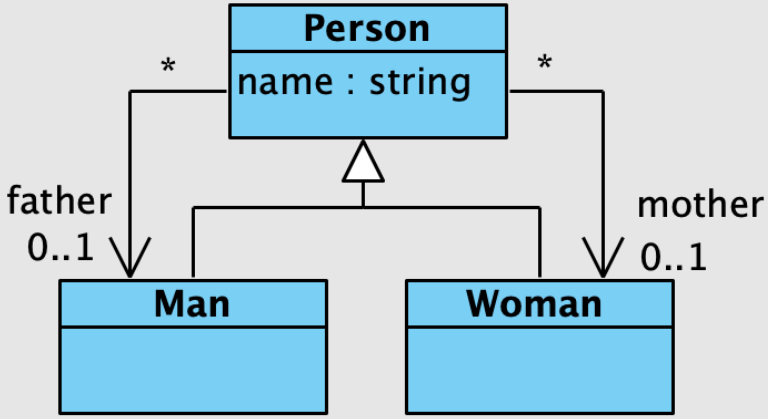
about the “recorded” world

Reference ontology

“real” world

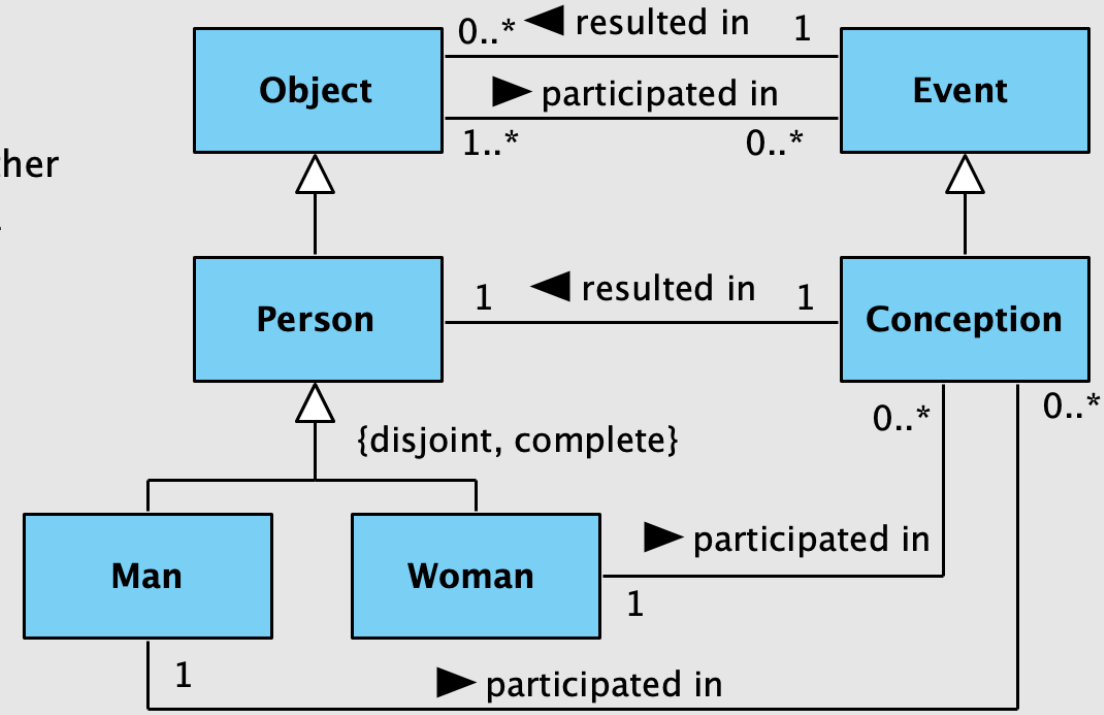
# Syntax, Semantics and All that Stuff

## Theory of the syntax




## Theory of the subject domain

domain-independent notions




Language metamodel

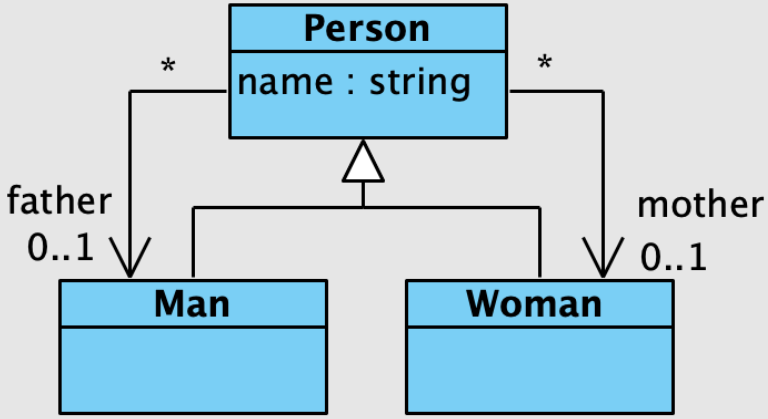
about the “recorded” world

Reference ontology

“real” world

# Syntax, Semantics and All that Stuff

## Theory of the syntax

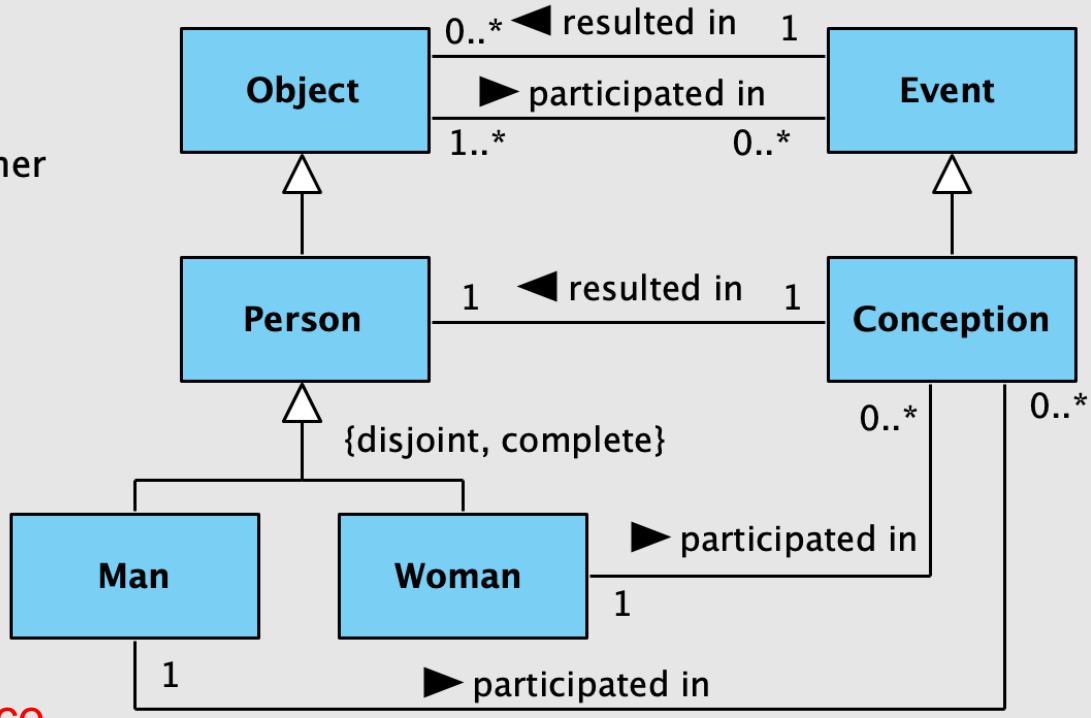



semantically-motivated syntactic constraints:
no ancestor cycles

no syntactic constraint to enforce ancestorOf transitive!

## Theory of the subject domain

domain-independent notions




ancestorOf acyclic, transitive

## Language metamodel

about the “recorded” world

## Reference ontology

“real” world

# Vision

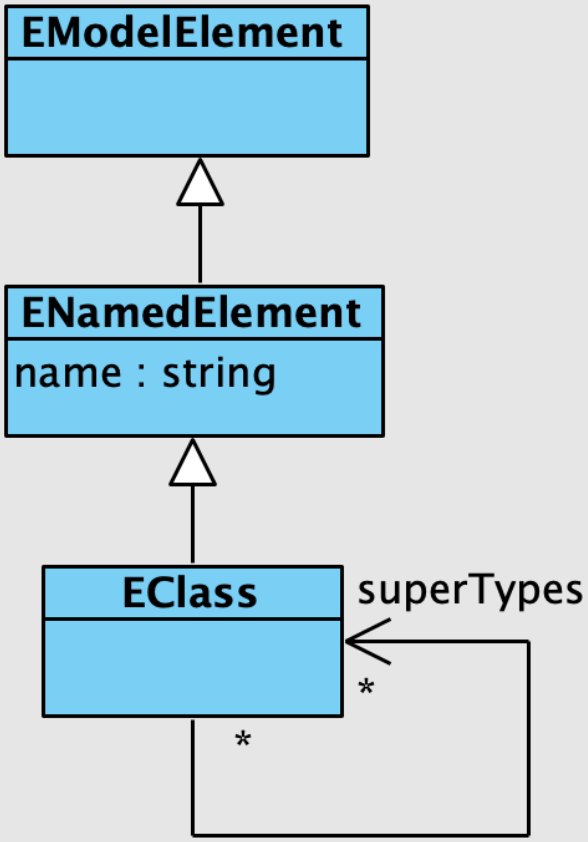
- Design of a language's semantics should be approach in a systematic way
- Including modeling of the language's domain on discourse
  - Not an arbitrary mathematical semantic domain
  - But one that can be used for meaning negotiation
  - As a formal artifact in our social game
  - Can be reused, integrated

- Exemplified for DSL but also applicable to general-purpose language

# Syntax, Semantics and All that Stuff

Theory of the syntax

Theory of the subject domain

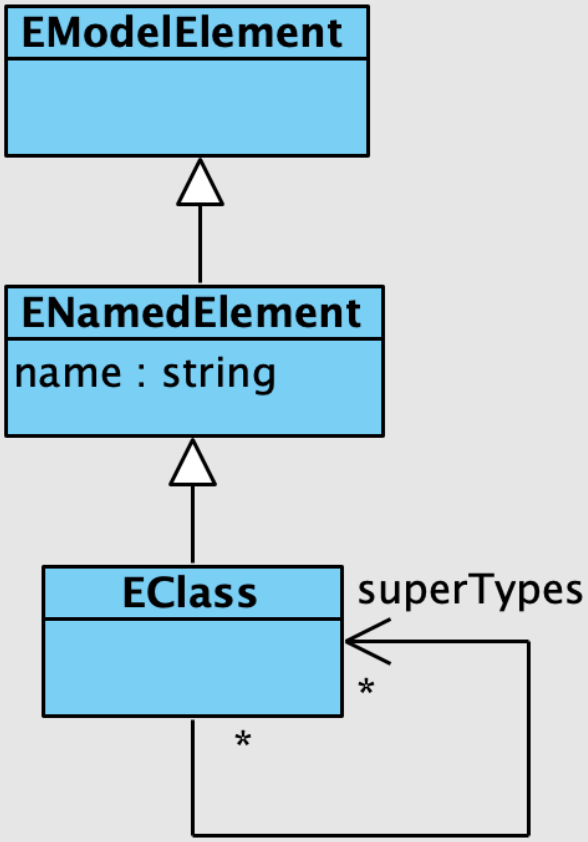



Language metamodel

Reference ontology

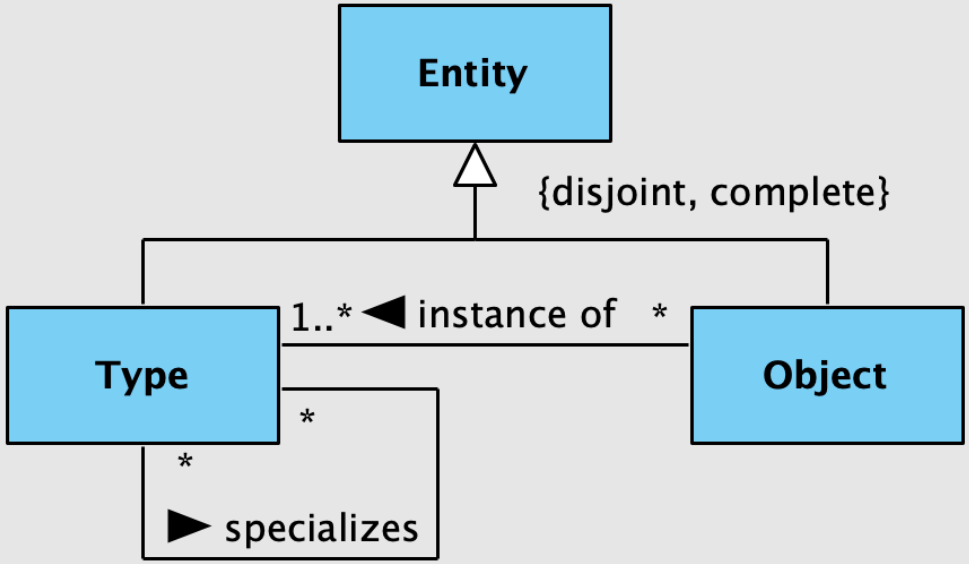
# Syntax, Semantics and All that Stuff

Theory of the syntax




Language metamodel

Theory of the subject domain

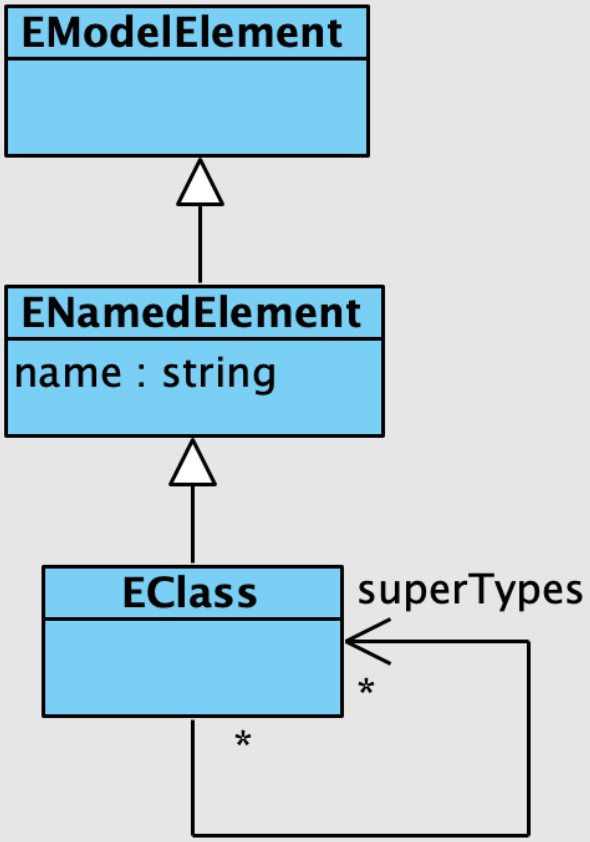



Settle semantic questions:
What is the meaning of specialization?
Can objects change type?
Can objects have more than one type with no common supertype?

Reference ontology

# Syntax, Semantics and All that Stuff

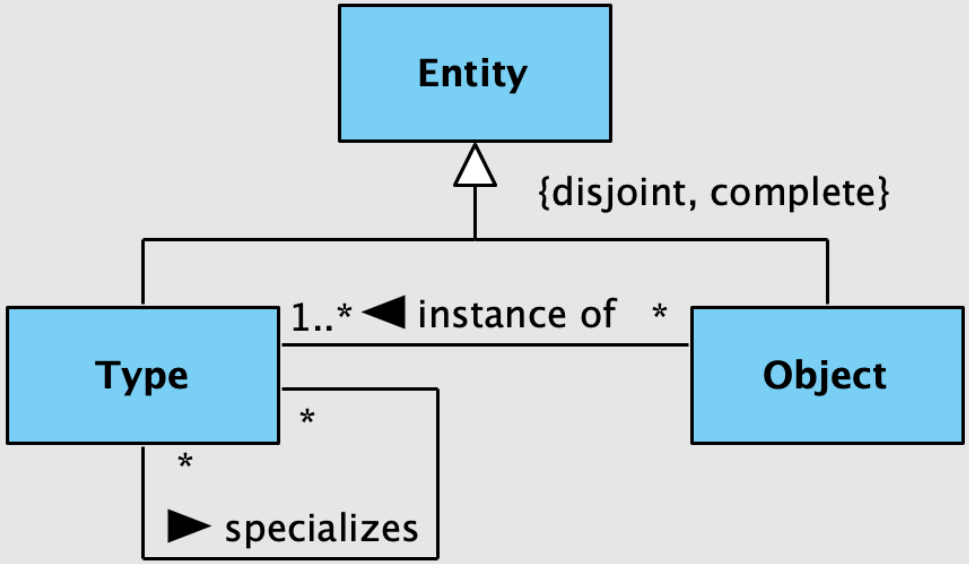
Theory of the syntax




syntactic constraint to enforce **specializes** acyclic!

Language metamodel

Theory of the subject domain

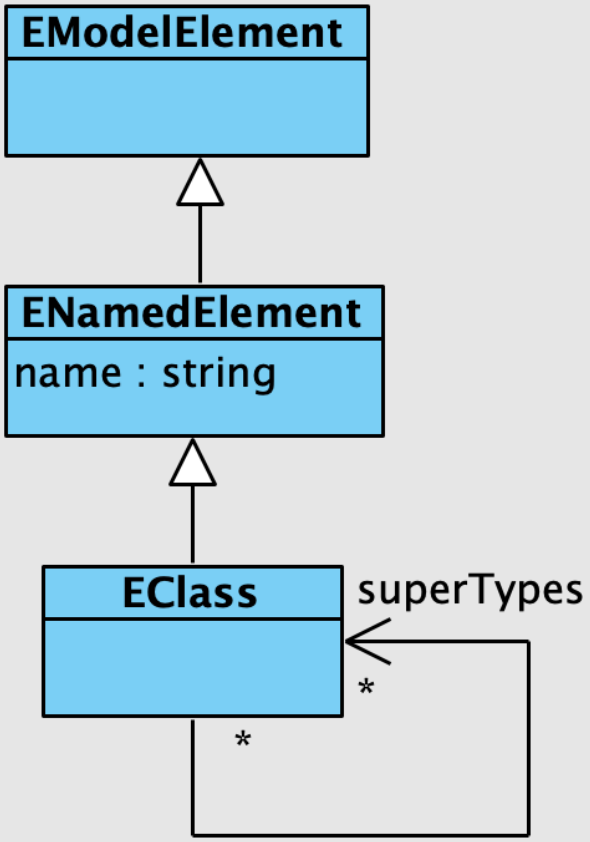



**specializes** is acyclic in virtue of its definition

Reference ontology

# Syntax, Semantics and All that Stuff

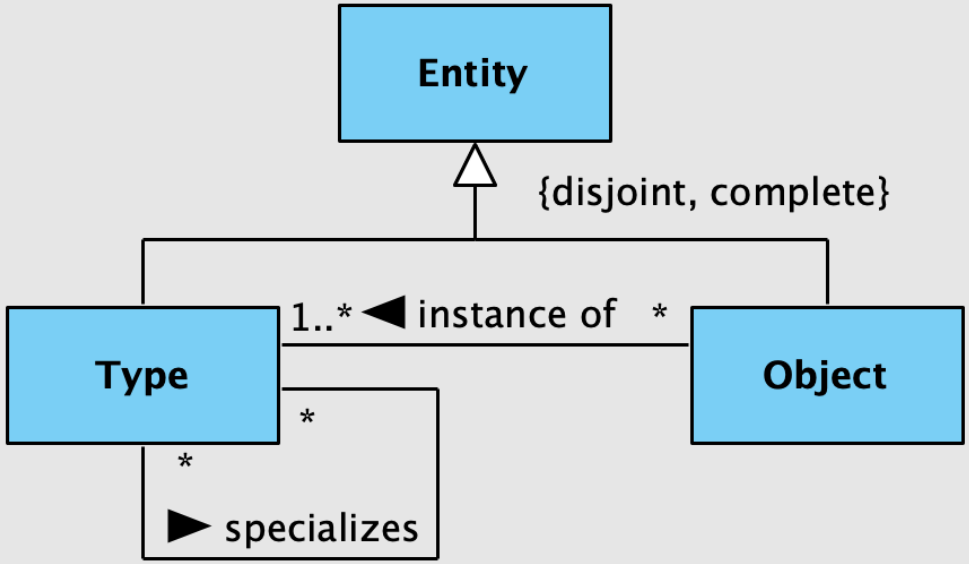
Theory of the syntax




no syntactic constraint to enforce **specializes** transitive!

Language metamodel

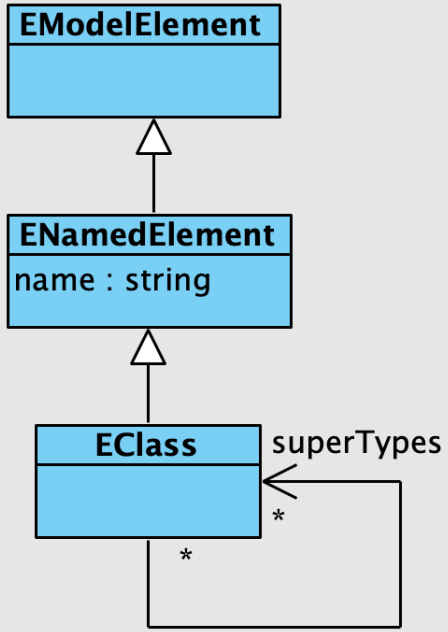
Theory of the subject domain




**specializes** is transitive in virtue of its definition

Reference ontology

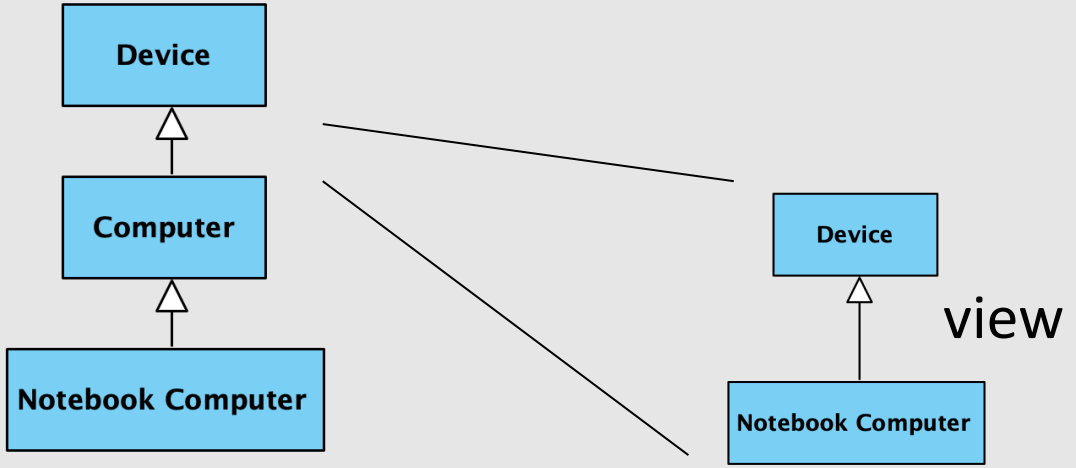
# View informed by semantics




SUM
Metamodel

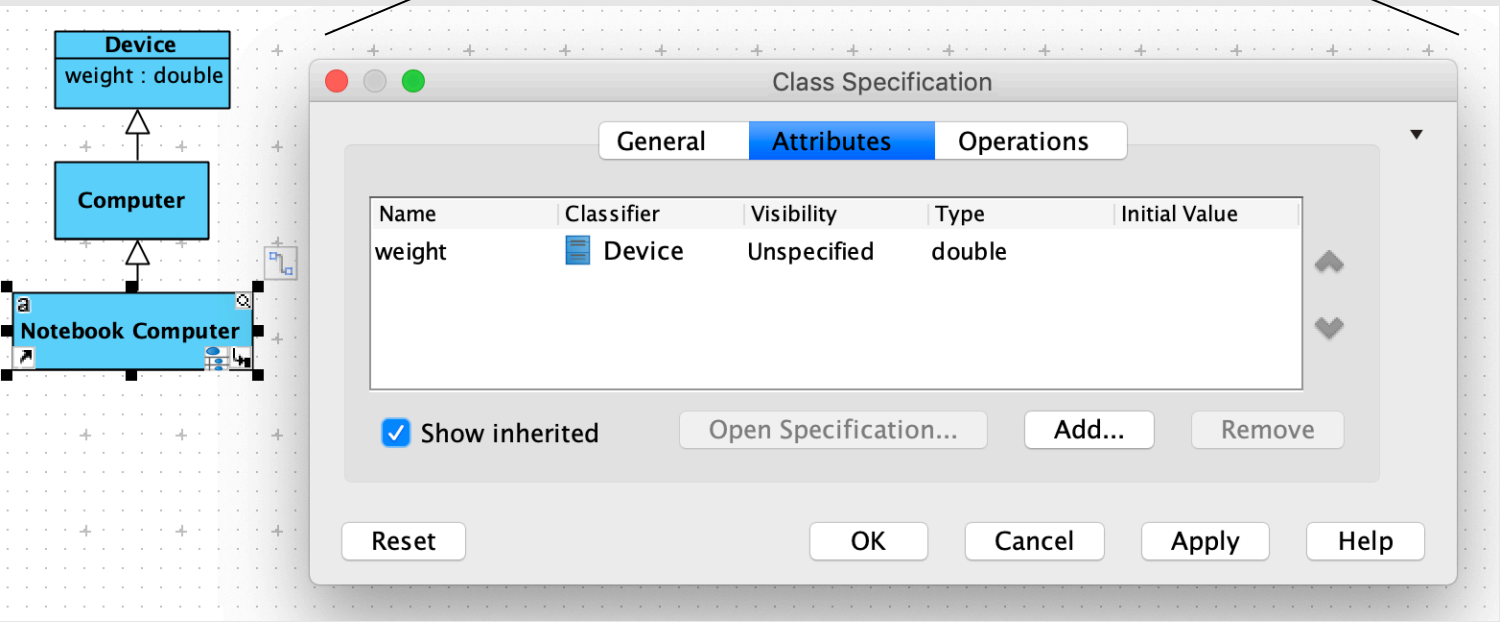
syntactic constraint to enforce **no** cycles

SUM

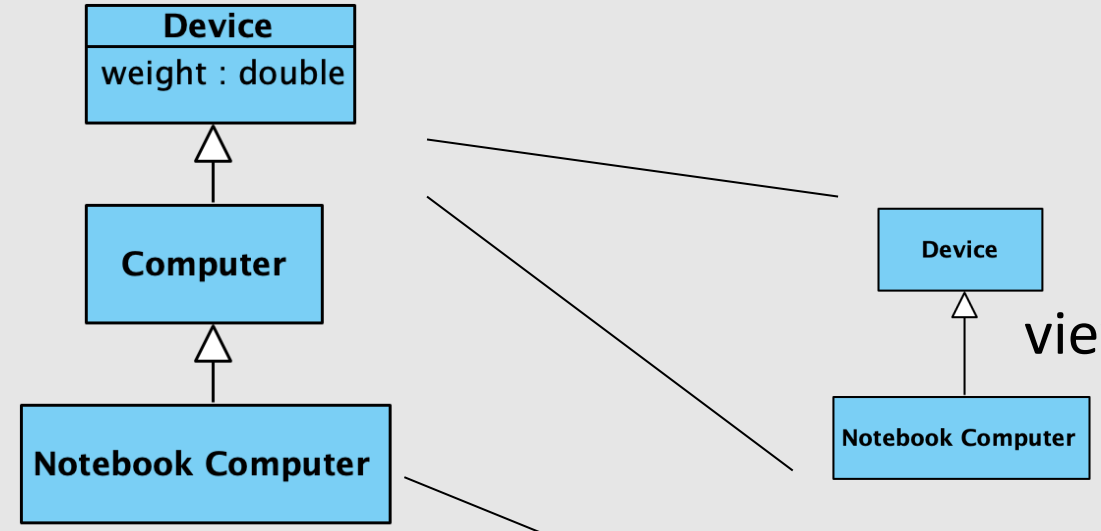



view

# View informed by semantics




SUM



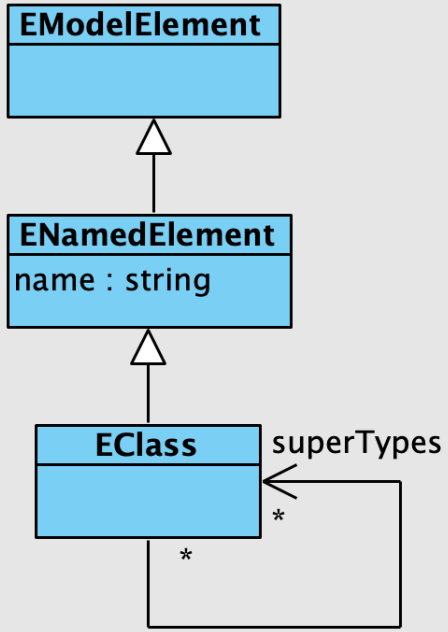




view 1

view 2

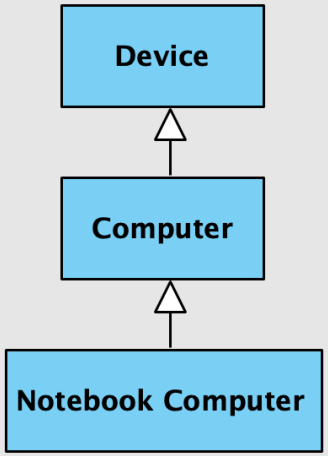
# SUM update informed by semantics



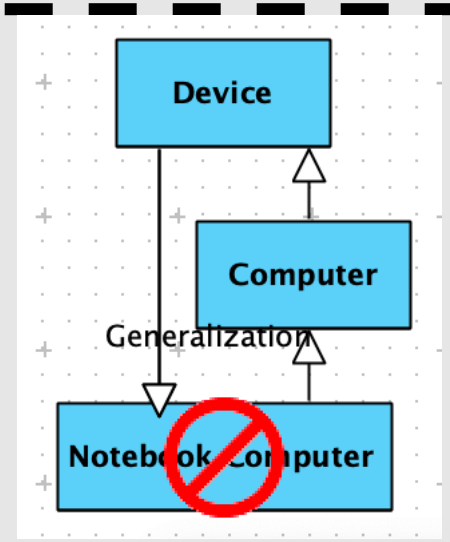

SUM Metamodel

syntactic constraint to enforce **no** cycles

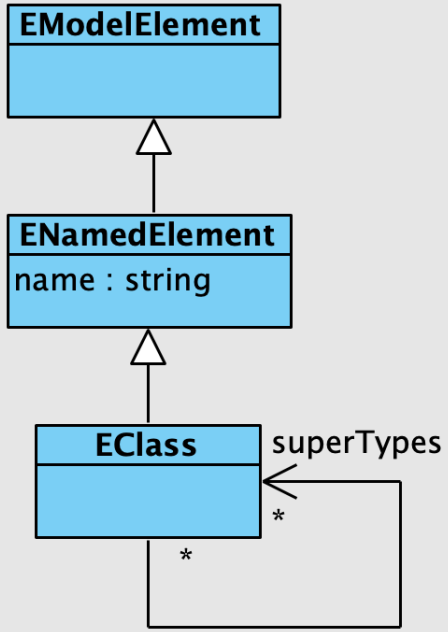
SUM




user wants update



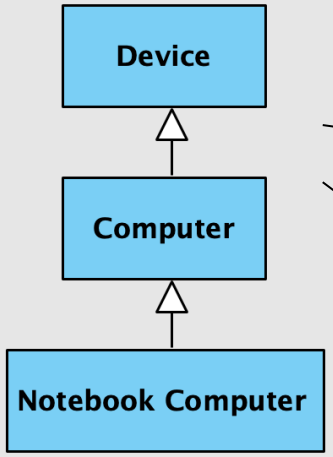
# View update informed by semantics



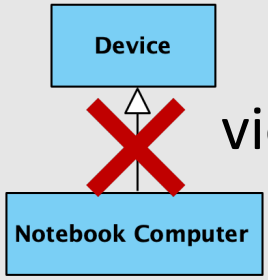

SUM Metamodel

syntactic constraint to enforce **no** cycles

SUM

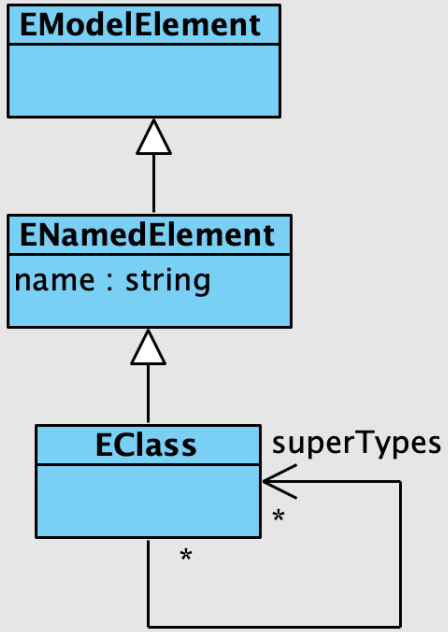



**multiple deletes possible In SUM**




view

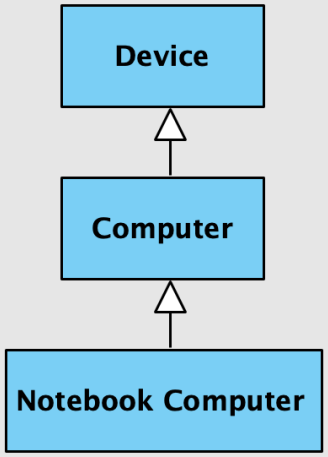
# Essential SUM metamodel?



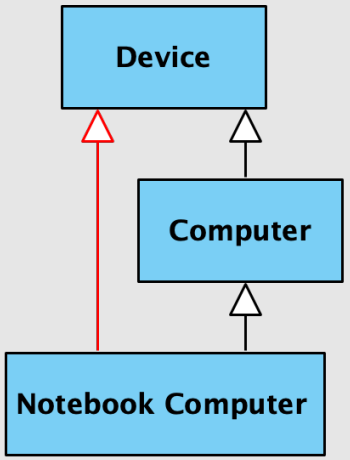

SUM Metamodel

syntactic constraint to enforce **no** cycles

SUM



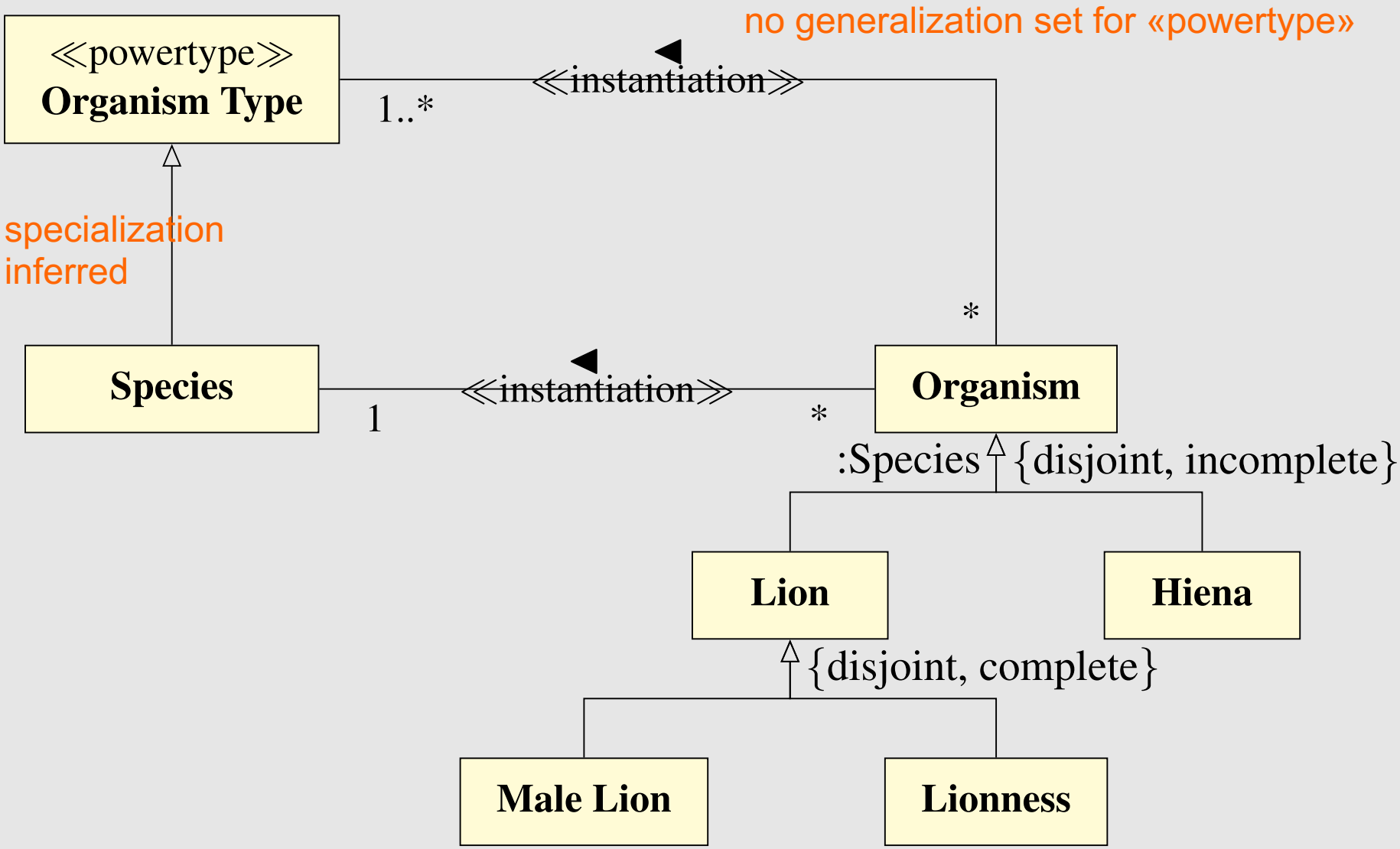

user wants update



# Essential SUM

- Entailment required to keep the SUM essential
- “Essential SUM” not “Essential SUM metamodel”
- Redundancy arises out of semantic notions not syntactic redundancy!

# Support for Cardelli's powertype



# Conclusions

- We use symbols to represent phenomena
- We should care about the phenomena they represent

- There is no “non ontology”, there is “bad ontology”
- Not doing “semantics” explicitly just means it gets ad hoc treatment

- Lightweight ontologies cannot do the trick by themselves
  - Very hard to know what is lost in the implementation
  - Need what is lost to drive (view-based) modeling
- Reference ontologies can help
  - But design of a language/view system/knowledge base is a different task

# Conclusions

- We need all the help we can get!
  - Ontology validation (e.g. simulation, anti-pattern detection)
  - Ontology verification
  - Ontology visualization
  - Reuse (e.g., of foundational ontology)

- Not a general-purpose reasoner, but designed (view-based) strategy:
  - Metamodels with semantically-motivated constraints
  - Semantically-motivated projective views
  - SUM and view update semantics informed by ontology

# Work inspired by this vision

- Ontology-driven conceptual modeling


- Foundational ontology
  - Unified Foundational Ontology (UFO)
    with Giancarlo Guizzardi and other NEMO members

- Multi-level modeling



UML powertypes

Wikidata

- Enterprise (architecture) modeling

ArchiMate

ARIS