(SysML 2/SST)
Semantics with a Little Math

Conrad Bock
U.S. National Institute of Standards and Technology

Ed Seidewitz
Model Driven

NIST
National Institute of Standards and Technology

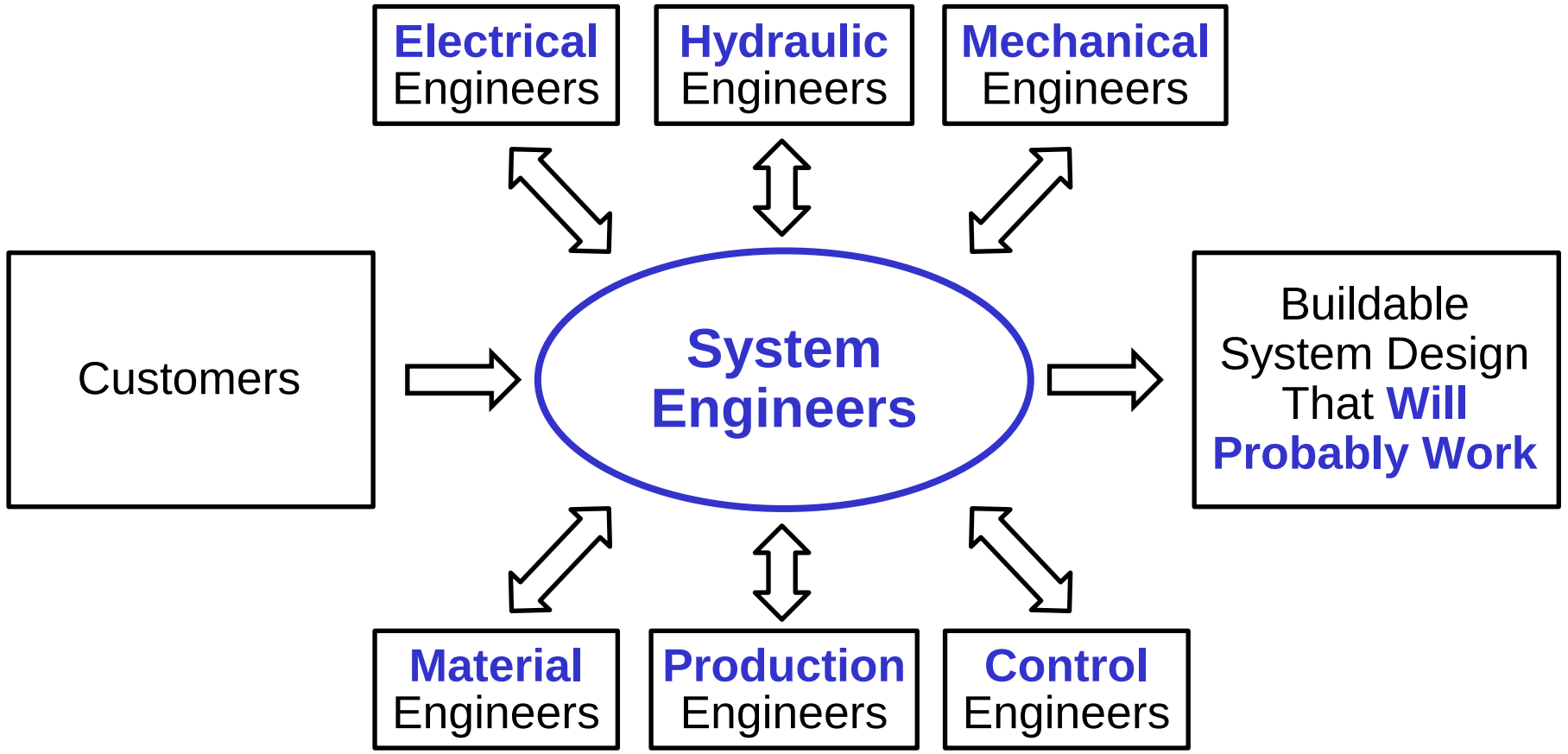
Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- Solution
  - The “S” Words
  - Standardizing Semantics
  - Conformance = Classification
  - Formalizing Semantics (ie, a little math)
  - SysML 2 Semantics
- The “O” Word
- Summary

Overview

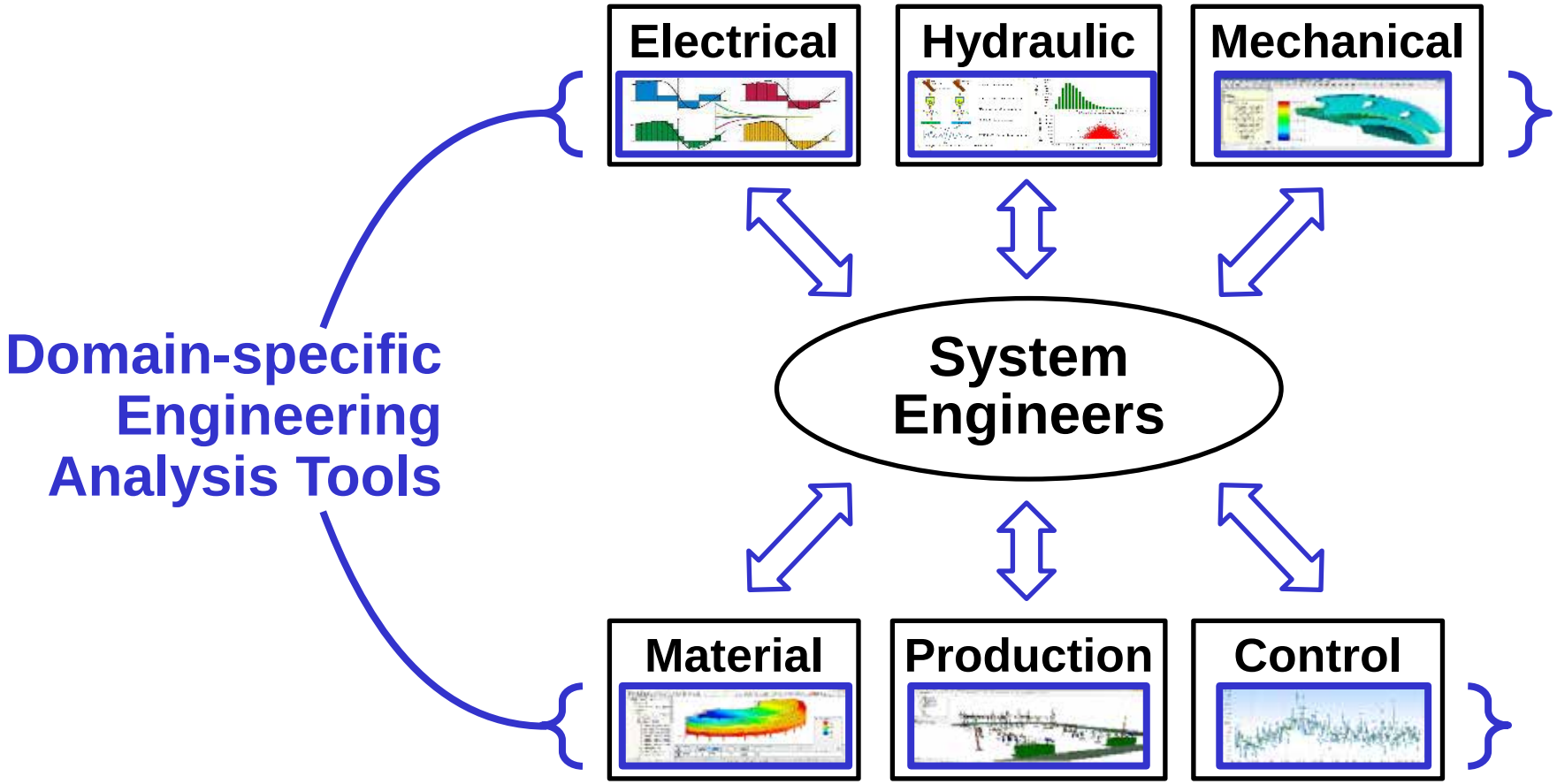
- **Motivation / Problem : Analysis**
  - **Systems Engineering**
  - **Modeling Languages**
- **Solution**
  - **The “S” Words**
  - **Standardizing Semantics**
  - **Conformance = Classification**
  - **Formalizing Semantics (ie, a little math)**
  - **SysML 2 Semantics**
- **The “O” Word**
- **Summary**

System Engineers



- **System engineers coordinate with all the other engineers**
  - **To produce a manufacturable design that will probably work.**

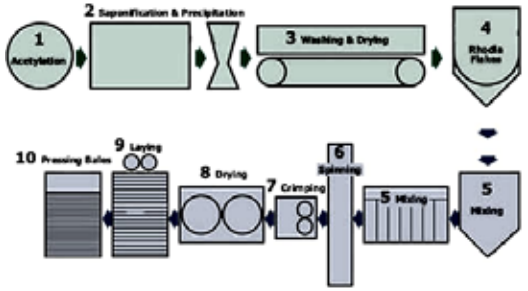
Getting to “Will Probably Work”



- **Domain engineers have mathematical tools for predicting how systems will behave.**

Engineering Analysis

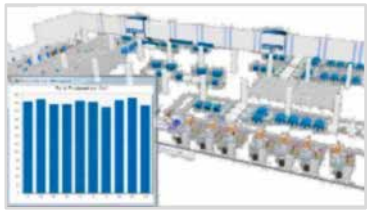
System model



Analyzing a Model

Building to a Model

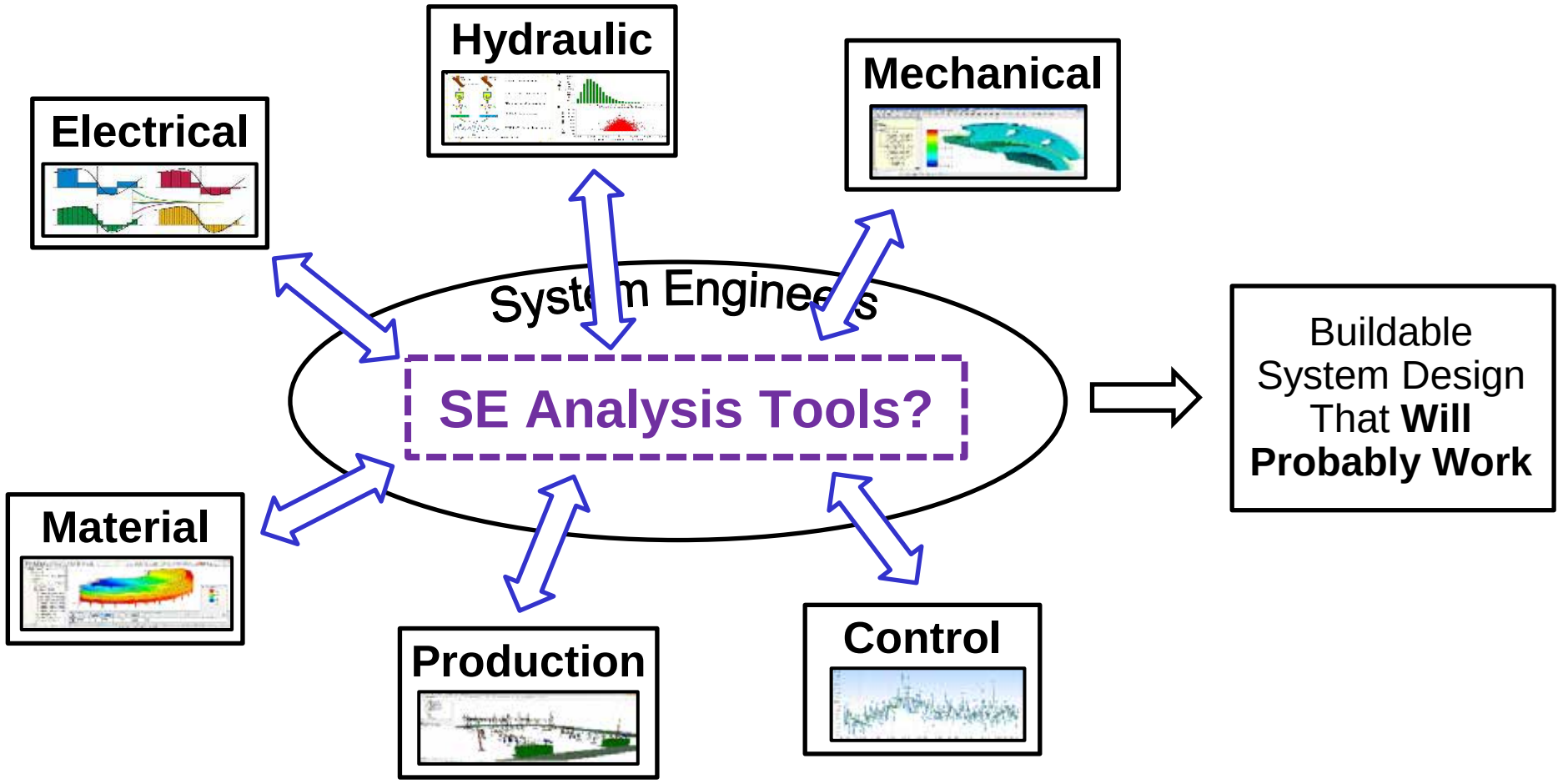
Real or Virtual Things being specified
(as simulated, constructed,operated)



predicts

- **Analyzers "imitate in advance" how real systems will be constructed, operated, and behave.**

SE and Engineering Analysis?



- **SEs need their own analysis tools to compare predicted behavior with domain engineers.**

Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - **Modeling Languages**
- Solution
  - The “S” Words
  - Standardizing Semantics
  - Conformance = Classification
  - Formalizing Semantics (ie, a little math)
  - SysML 2 Semantics
- The “O” Word
- Summary

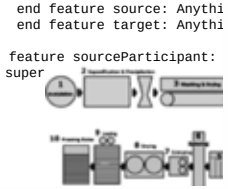
Modeling

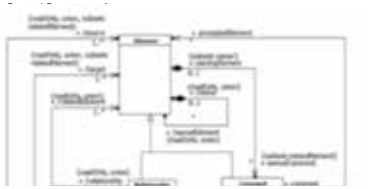
Language Developers

(using *example models*)

```
    assoc BinaryLink specializes Link {
feature participant: Anything[2] nonunique
 end feature source: Anythi
 end feature target: Anythi

feature sourceParticipant:
super
```

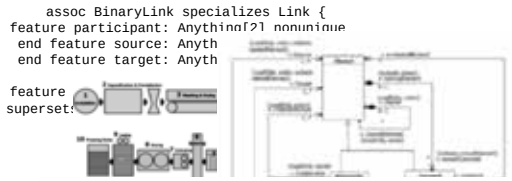




What are they imagining for system operation?

Modeling

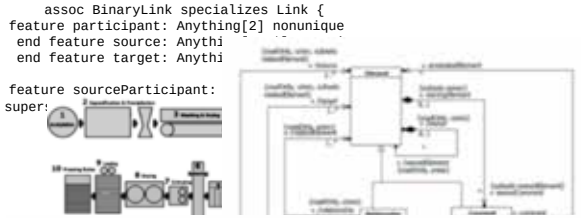
Language Developers
(using *example models*)



What are they imagining for system operation?

Analysis

Analysis Tool Builders
(incl execution, simulation, reasoning, etc)



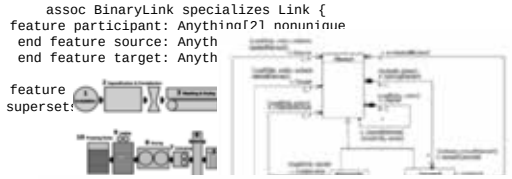
What should tools predict for system operations?

Modeling and Analysis

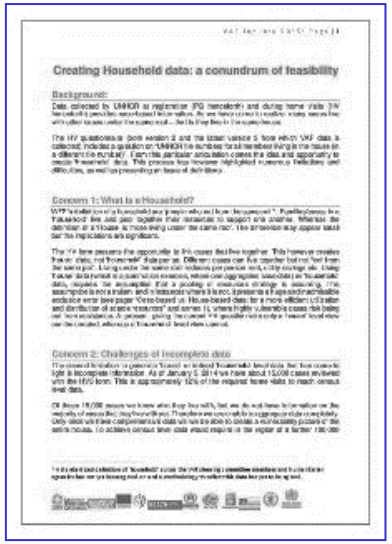
Don’t know each other

Communicate only
through a standards spec

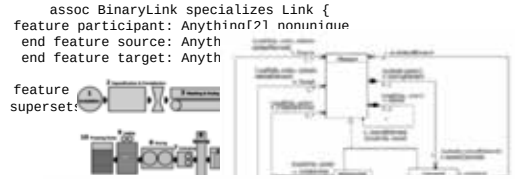
Language Developers
(using *example models*)



What is imagined for system operation?



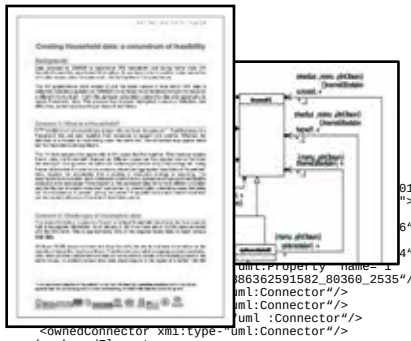
Analysis Tool Builders
(incl execution, simulation, reasoning, etc)



What should tools predict for system operations?

Modeling Language Lifecycle

Standards definers



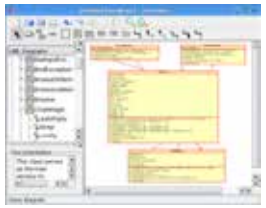
```
<ownedConnector xmi:type-"uml:Connector"/>
</packagedElement>
<packagedElement xmi:type-"uml:Class" name="Source"/>
<packagedElement xmi:type-"uml:Class" name="Ground"/>
<packagedElement xmi:type-"uml:Class" name="Capacitor"/>
<packagedElement xmi:type-"um1:Association"/>
<packagedElement xmi:type="um1:Association"/>
</uml:Model>
</xmi>
```

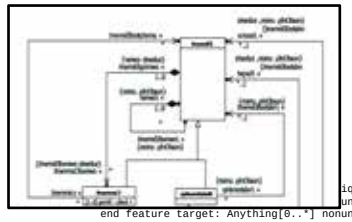
Tool builders

Tool users

System builders

Modeling

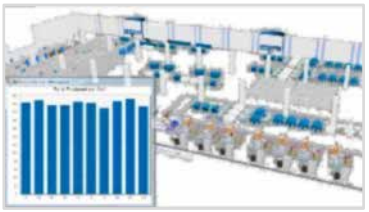


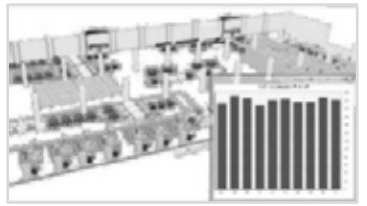


```
                                                                    ique
                                                                    unique
end feature target: Anything[0..*] nonunique

feature sourceParticipant: Anything[1] subsets
supersets targetParticipant;  // pairs of links
```

Analysis





Construction

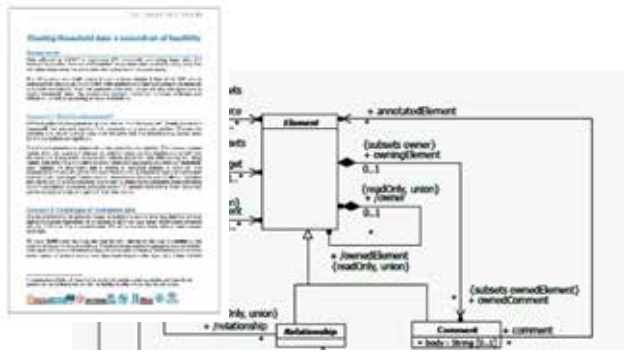
Control & Mointoring

Effect

Feedback

Modeling Languages, Part 1

Information model / modeling language standard



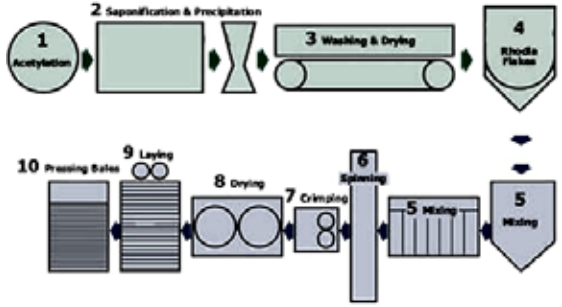
Specifies models

1

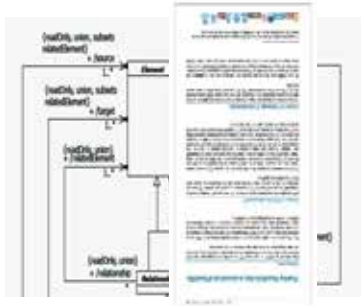
Using a Modeling Language

- § **Graphics**:
  - – Circles
  - – Lines
  - – Rectangles
- § **Text**
  - – Reserved
  - – Order
- § **Linguistic terms**:
  - – Blocks, Item Flows
  - – Activity, Interaction
- § **Using terms**:
  - – Make an activity the behavior for a block

System model

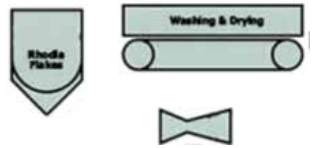


reuses



Domain library

- § **Domain graphics**:



- § **Text**
  - – Reserved
  - – Order
- § **Domain terms**:
  - – Lathes, Feeders
  - – Drying, Shaping
- § **Using terms**:
  - – Connect a feeder to a lathe

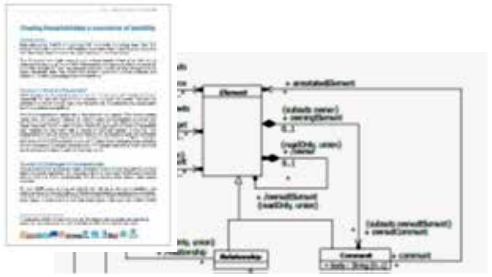
§ **Modeling tools follow a language standard**

– **Often using (standard) model libraries**

Modeling Languages, Parts 2 & 3

Specifies models ①

Information model / modeling language standard

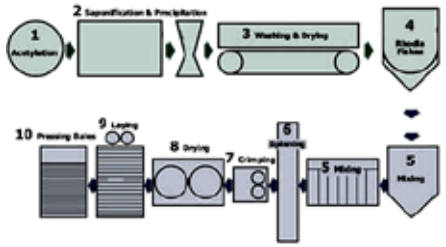


- § **Graphics:**
  - – Circles
  - – Lines
  - – Rectangles
- § **Text**
  - – Reserved
  - – Order
- § **Linguistic terms:**
  - – Blocks, Item Flows
  - – Activity, Interaction
- § **Using terms:**
  - – Make an activity the behavior for a block

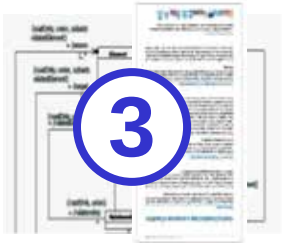
Using a Modeling Language

Specifies real or virtual things ②

System model



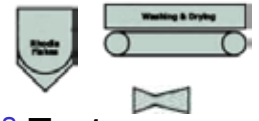
reuses



③

Domain library

- § **Domain graphics:**



- § **Text**
  - – Reserved
  - – Order
- § **Domain terms:**
  - – Lathes, Feeders
  - – Drying, Shaping
- § **Using terms:**
  - – Connect a feeder to a lathe

Analyzing a Model

Building to a Model

Real or Virtual Things being specified
(as simulated, constructed,operated)

predicts

Example: Natural Language

Specifies sentences

1

Grammar

Sentence
- Noun Phrase
  - Determiner
  - Noun
- Verb Phrase
  - Verb
  - Noun Phrase

§ Alphabet:
§ Word order
§ Kinds of words
§ Using words

Using a Natural Language

Specifies real or virtual things

2

Sentences

"A quick brown fox jumps over a lazy dog."

reuses

3

Vocabulary

§ Nouns
– mountain
– road

§ Verbs
– climb
– drive

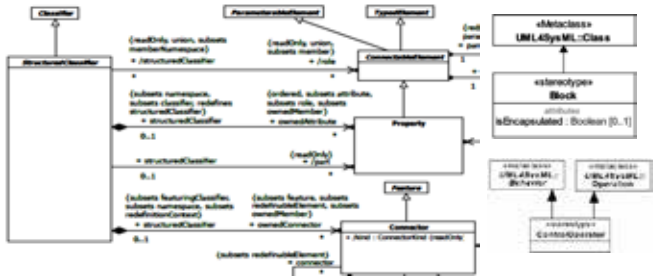
Describes

Real or Virtual Things being described

No nouns and verbs here

Example: SysML/UML

UML Metamodel / SysML stereotypes



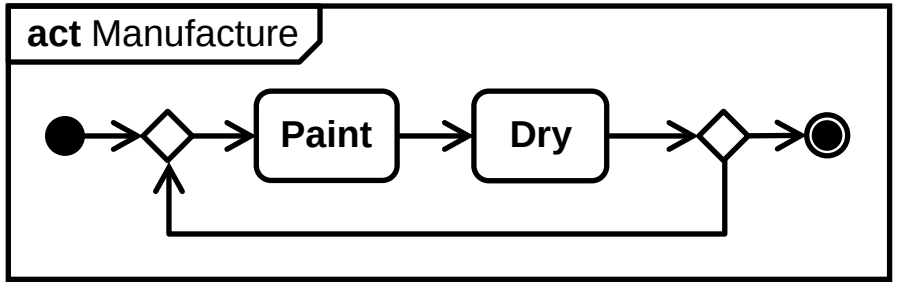
- § Blocks
- § Activities
- § Actions
- § Control Flow

Specifies SysML models (1)

Using SysML

Specifies real or virtual things (2)

System Model



reuses

(3)

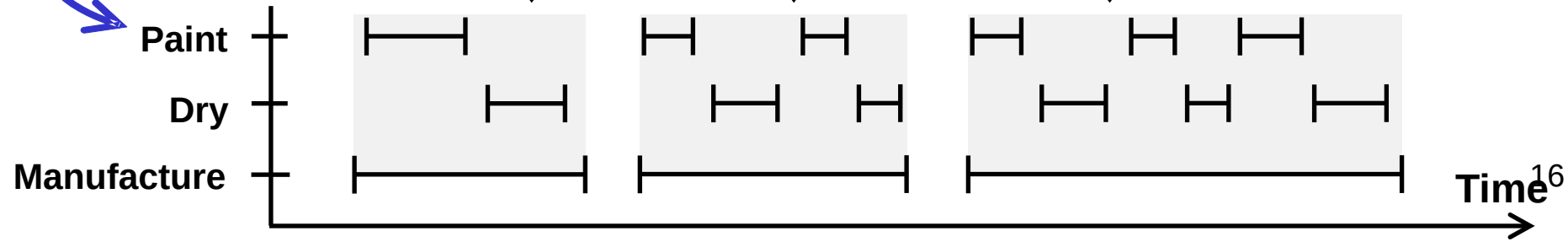
- § Behaviors
  - – Paint
  - – Dry

Domain library

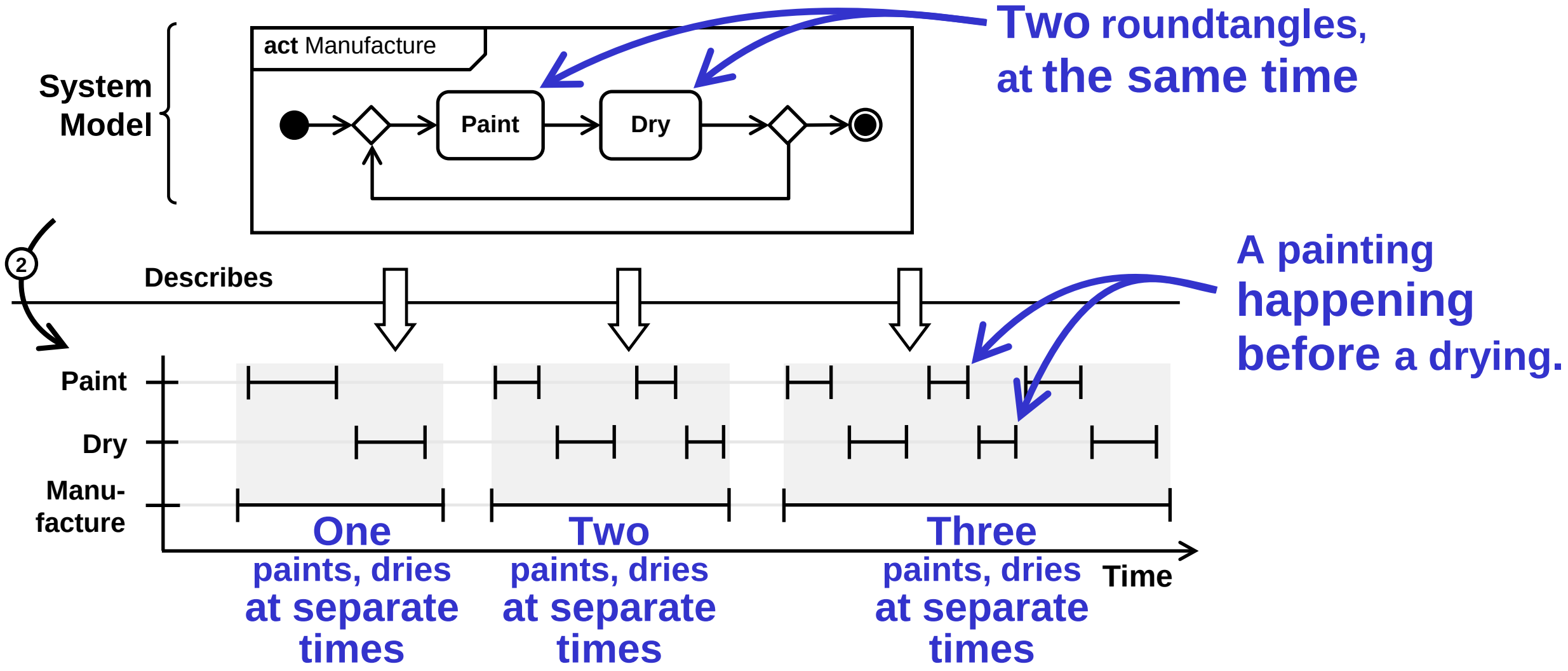
Describes

Real or Virtual Things being described

No blocks and flows here



Models ≠ Things Being Modeled



§ **Models are not in time** (compare to model versioning)

No Blocks on the Tarmac

Where are the blocks? Item flows?

OMGer

Pilot

The “L” Word

- “Language”
- Usually interpreted as = vocabulary
  - Spoken/written words, eg, “plane”, “bonjour”, etc.
- In software/OMG circles = reserved words
  - Words defined in standard, eg, “block”, “if then”, etc.
  - Reserved = can only be used as specified.
  - Not vocabulary in the usual sense.
- Formal language theory
  - Coming up!

A Map is not the Territory

Contour lines
(constant elevation)

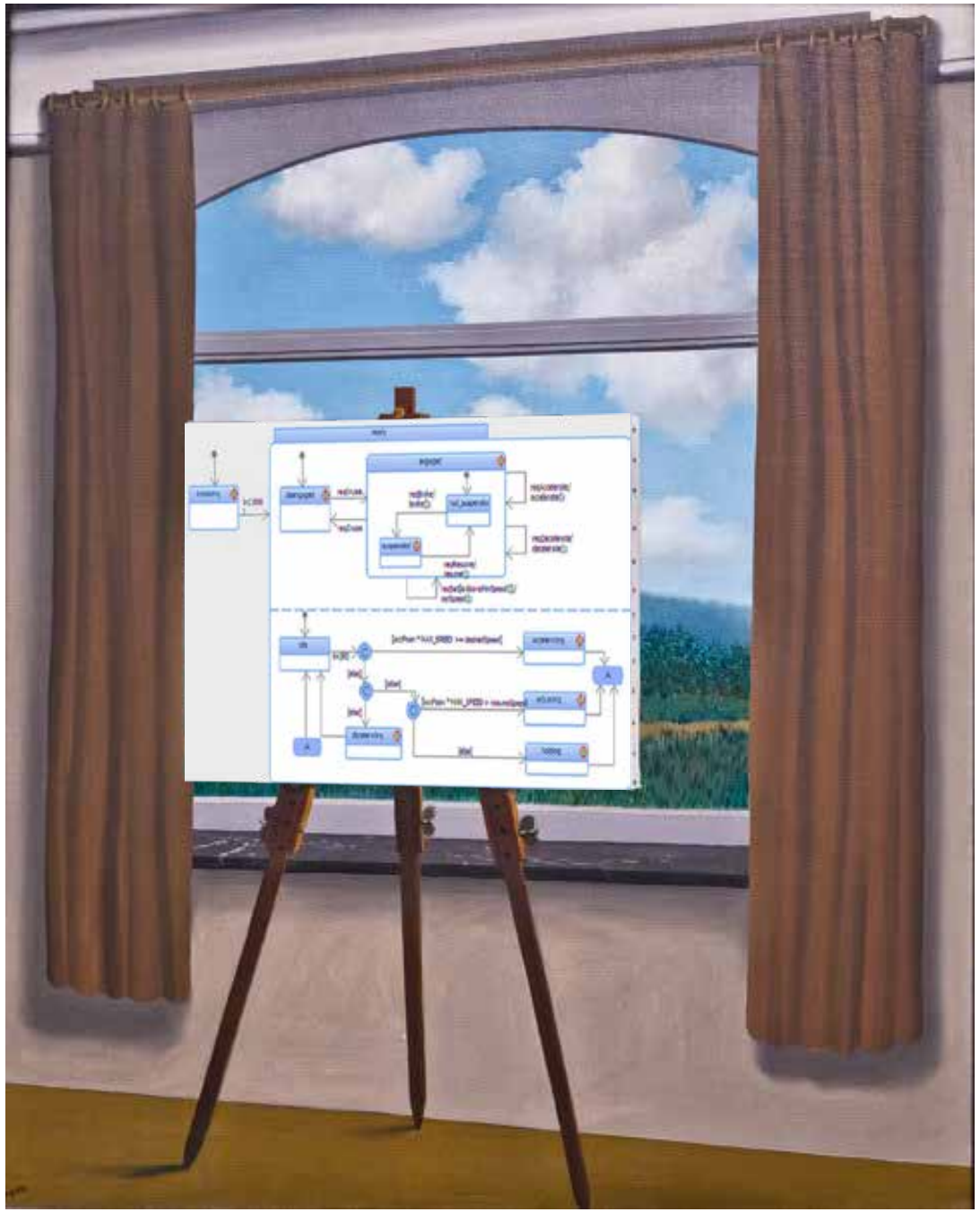
No contour lines

https://en.wikipedia.org/wiki/Alfred_Korzybski

Magritte

https://en.wikipedia.org/wiki/The_Human_Condition_(Magritte)

Magritte at OMG



If You Don’t Get This …

- … it’s OK!
- Only a small number of computer folks do
  - They write compilers for programming languages
- It’s capital equipment
  - For producing useful analysis software
- But the rest of this presentation might be confusing
  - and it might not be clear why systems engineering languages can’t succeed without it, because …
  - … SEs won’t be able to interact with domain-specific engineers, who all use analysis tools.

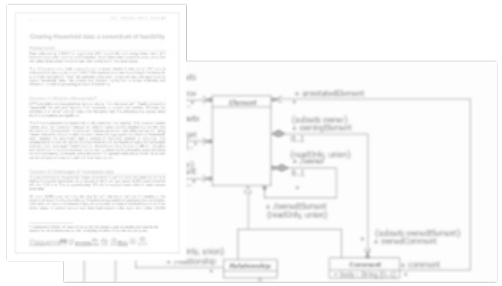
Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- **Solution**
  - **The “S” Words**
  - Standardizing Semantics
  - Conformance = Classification
  - Formalizing Semantics (ie, a little math)
  - SysML 2 Semantics
- The “O” Word
- Summary

Technical Terms for Parts 1 & 2

Syntax
specifies models
1

Information model / modeling language standard

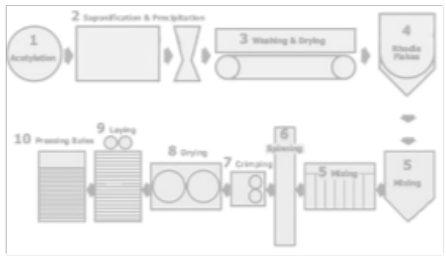


- Graphics:
  - Circles
  - Lines
  - Rectangles
- Text
  - Reserved
  - Order
- Linguistic terms:
  - Blocks, Item Flows
  - Activity, Interaction
- Using terms:
  - Make an activity the behavior for a block

Using a Modeling Language

Semantics
specifies real or virtual things
2

System model



reuses

Domain library

- Domain graphics:

- Text
  - Reserved
  - Order
- Domain terms:
  - Lathes, Feeders
  - Drying, Shaping
- Using terms:
  - Connect a feeder to a lathe

Analyzing a Model

Building to a Model

Real or Virtual Things being specified
(as simulated, constructed,operated)

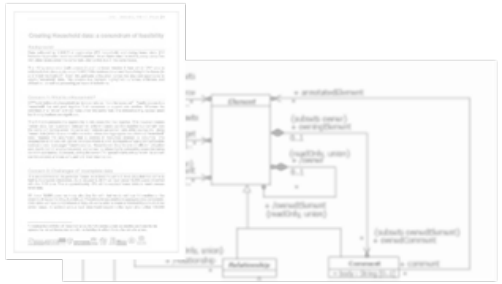
predicts

Inverse Terms for Parts 1 & 2

Models conform to syntax (or not) ① ↑

Real or virtual things conform to semantics (or not) ② ↑

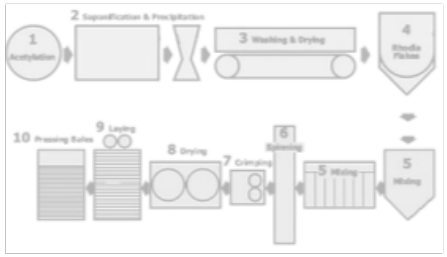
Information model / modeling language standard



- Graphics:
  - Circles
  - Lines
  - Rectangles
- Text
  - Reserved
  - Order
- Linguistic terms:
  - Blocks, Item Flows
  - Activity, Interaction
- Using terms:
  - Make an activity the behavior for a block

Using a Modeling Language

System model



reuses

Domain library

- Domain graphics:

- Domain terms:
  - Lathes, Feeders
  - Drying, Shaping
- Text
  - Reserved
  - Order
- Using terms:
  - Connect a feeder to a lathe

Analyzing a Model

Building to a Model

Real or Virtual Things being specified
(as simulated, constructed,operated)

predicts

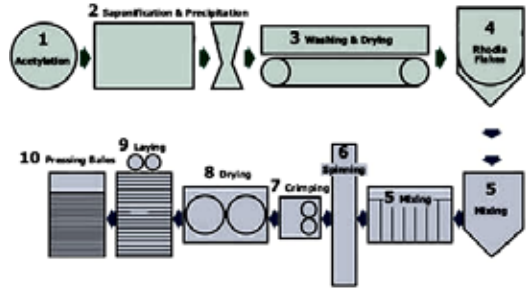
§ **Conformance is a yes/no question.**

The (second) “S” Word

- One meaning used here: how to tell when …
  - a real or virtual thing (as constructed and operated) …
  - “follows” (conforms to) a model …
  - … written in a particular language.
- “How to tell” =
  - procedure resulting in true or false when applied to real or virtual thing/operation.
  - Based on conditions that must be met by operated thing.
- Compare to
  - Application vocabulary (lathes, drills, etc).
  - Model development methods (requirements, designs).

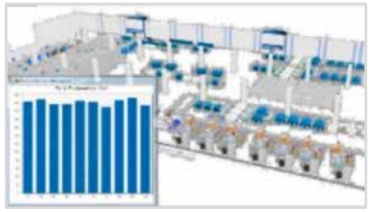
Checking Semantics

Engineering model



Interpreting a Model

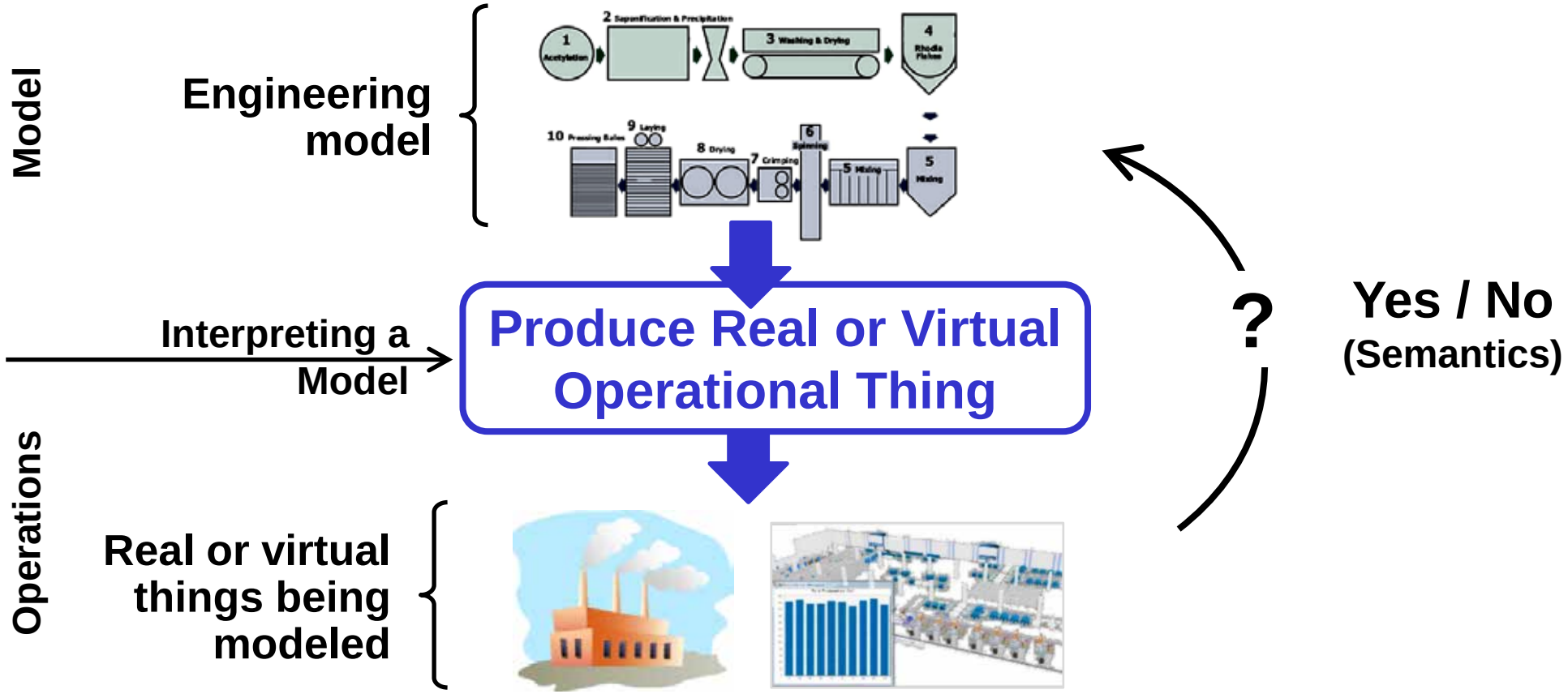
Things being specified (as constructed & operated)



? Yes / No (Semantics)

- How do we know whether real or virtual things built & operated to a model follow the model?
- = Semantics (a boolean check)

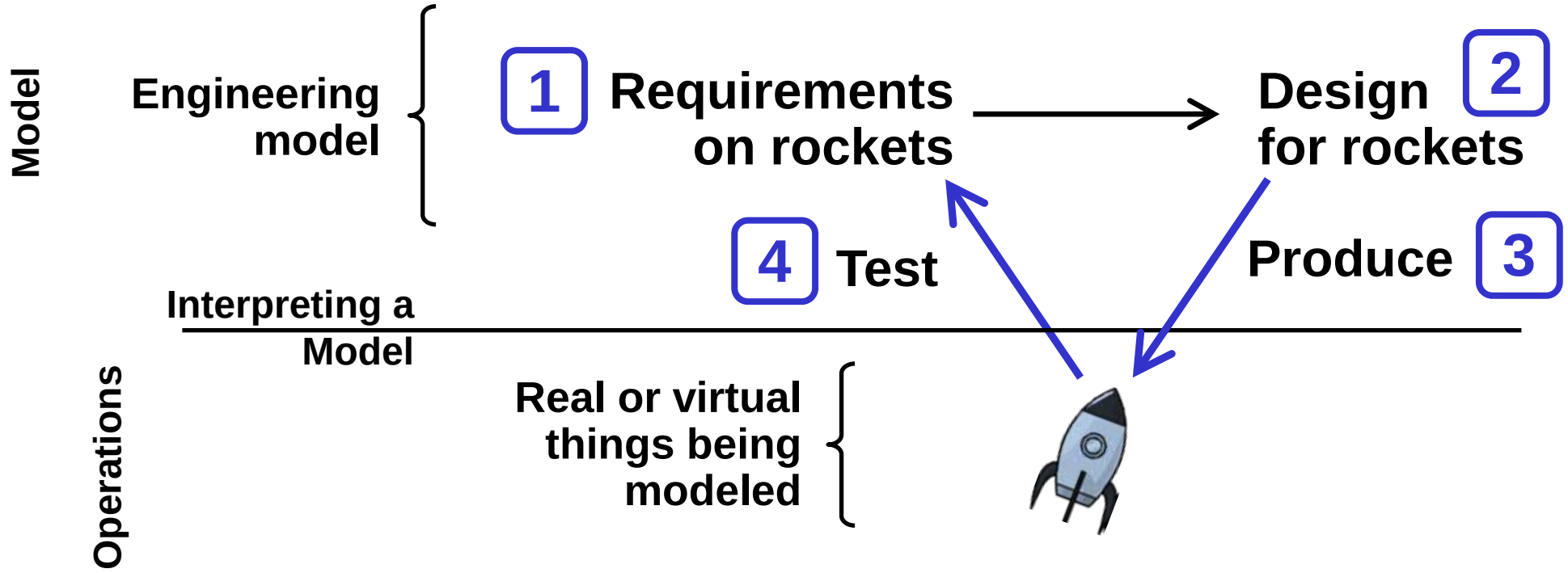
Producing Real/Virtual Things from Models



§ **Producing real/virtual things is difficult.**

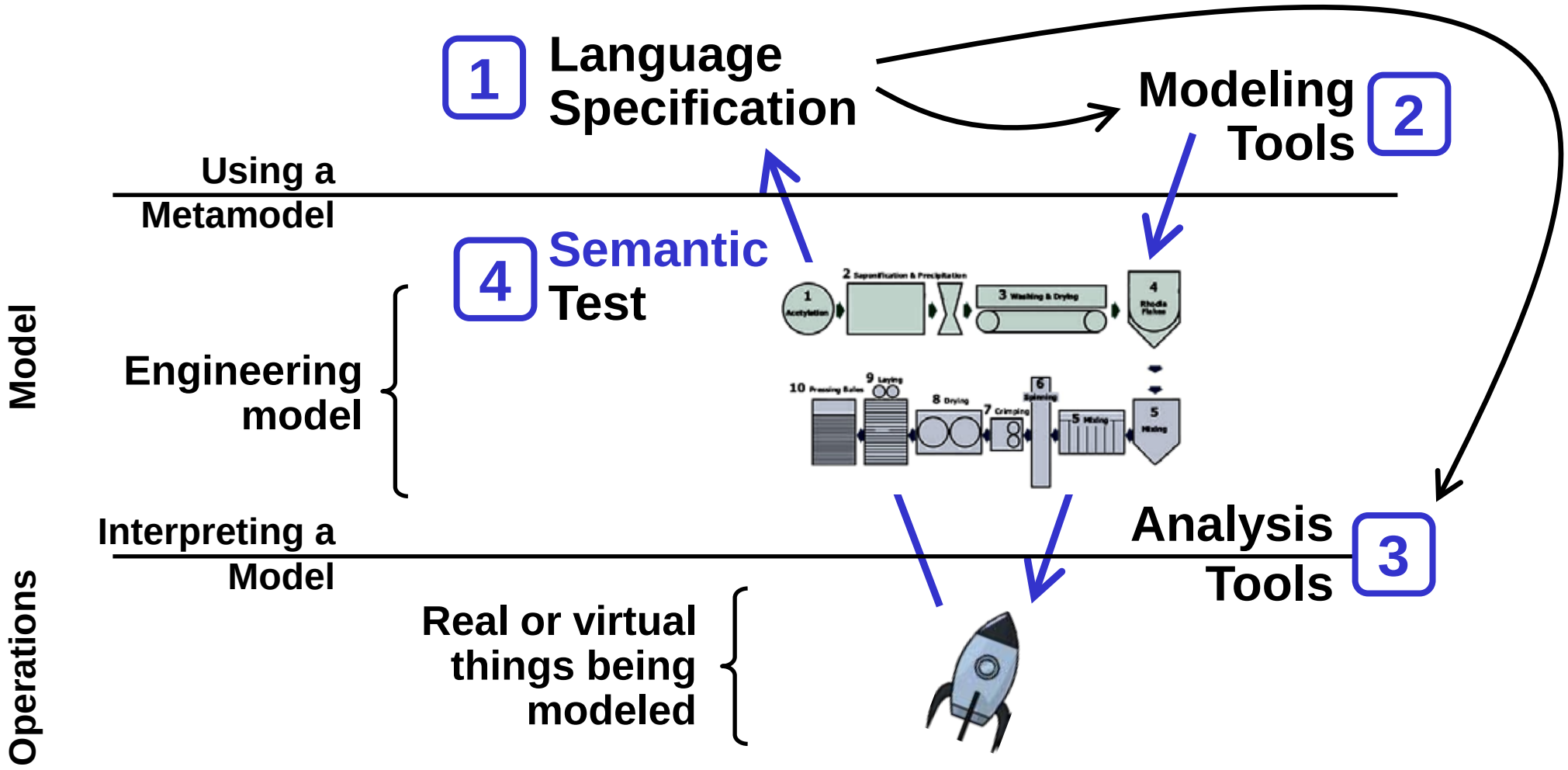
– **Checking these things (using semantics) is easy.**

SE: Requirements, Designs, Tests



- Do real/virtual systems meet requirements when **built and operated** according to a design?

Language Specs & Implementations



§ **Do analysis tools "meet" the language spec?**

Systems Engineering for Languages

- **SE involves multiple kinds of specifications:**
  - **Intended effects of a language (requirements ↔ semantics)**
  - **How models bring about effects (designs ↔ analysis tools)**
  - **Testing whether real/virtual systems have the required effects when built/operated per a model (tests ↔ semantic checking).**

Systems Engineering	Modeling Languages
Requirements	Semantics
Designs	Analysis Tools
Tests	Semantic Checking

SST: Standardize Checking, Not Production

- **Many ways to create and analyze models based on a standard language**
  - **Many ways to design a system to meet requirements**
- **OMG doesn't specify how to create models**
  - **Just how to interchange and access them (syntax/API).**
- **It shouldn't specify how to analyze them either**
  - **Just how to tell when results are correct (semantic check).**

Technical Term: Inference

- **Produce** real or virtual things from models = inference
  - Execution
    - Incremental creation, usually deterministic and time ordered.
  - Simulation
    - Less deterministic execution.
    - Aggregate measures of probable executions.
  - Reasoning
    - Search based directly on semantics.

Kinds of **inference** (logically speaking)

- Inference procedures are **evaluated** by whether results
  - Always pass semantic check (**soundness**).
  - Include everything that can pass it (**completeness**).
  - Can be produced (how) quickly (**complexity**).

See Section 3.1 (Intro to Reasoning) in Bock, et al, "Evaulating Resoning Systems," NIST 7310 https://www.nist.gov/publications/evaluating-reasoning-systems

Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- Solution
  - The “S” Words
  - **Standardizing Semantics**
  - Conformance = Classification
  - Formalizing Semantics (ie, a little math)
  - SysML 2 Semantics
- The “O” Word
- Summary

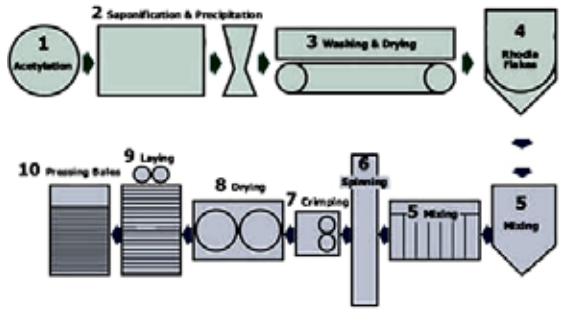
Standardizing Conformance, Syntactic

Information model / modeling language standard

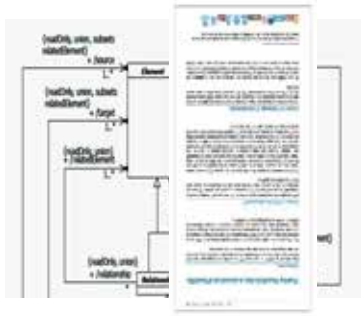
- § Graphics:
  - – Circles
  - – Lines
  - – Rectangles
- § Text
  - – Reserved
  - – Order
- § Linguistic terms:
  - – Blocks, Item Flows
  - – Activity, Interaction
- § Using terms:
  - – Make an activity the behavior for a block

Using a Modeling Language

System model



reuses



Domain library

Syntactic conformance

- § **Typical “instance checking”**
  - – **between metamodel and model**
  - – **specified in the usual way (classes, properties, constraints)**

Standardizing Conformance, Semantic ?

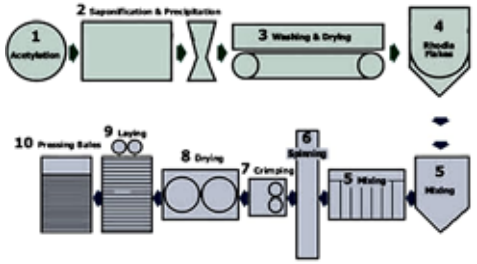
Information model / modeling language standard

- § Graphics:
  - – Circles
  - – Lines
  - – Rectangles
- § Text
  - – Reserved
  - – Order
- § Linguistic terms:
  - – Blocks, Item Flows
  - – Activity, Interaction
- § Using terms:
  - – Make an activity the behavior for a block

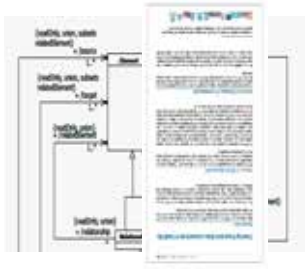
?

Using a Modeling Language

System model



reuses



Domain library

Interpreting a specification

Operations

Real or virtual things being modeled

Semantic conformance

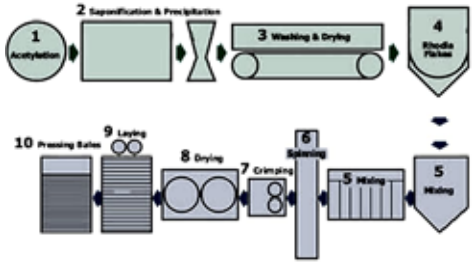
Checking Semantic Conformance, Manual

Modeling language standard

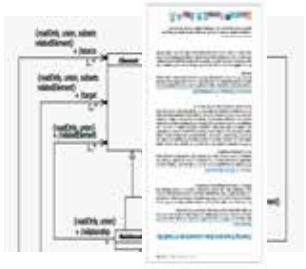
- § Graphics:
  - – Circles
  - – Lines
  - – Rectangles
- § Text
  - – Reserved
  - – Order
- § Linguistic terms:
  - – Blocks, Item Flows
  - – Activity, Interaction
- § Using terms:
  - – Make an activity the behavior for a block

Using a Modeling Language

System model



reuses



Domain library

Interpreting a specification

Operations

Real or virtual things being modeled

Yes / No

Semantic conformance checked *manually* based on free text

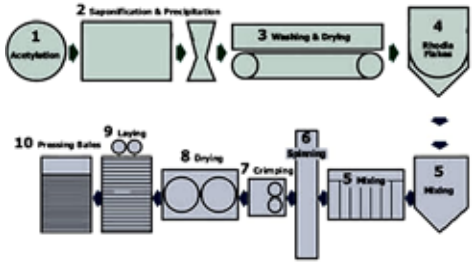
Checking Semantic Conformance, Autoish

Modeling language standard

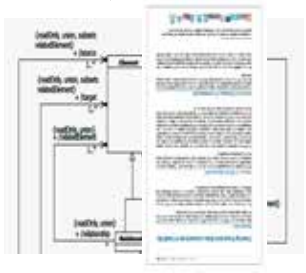
- § Graphics:
  - – Circles
  - – Lines
  - – Rectangles
- § Text
  - – Reserved
  - – Order
- § Linguistic terms:
  - – Blocks, Item Flows
  - – Activity, Interaction
- § Using terms:
  - – Make an activity the behavior for a block

Using a Modeling Language

System model



reuses



Domain library

Interpreting a specification

Operations

Real or virtual things being modeled

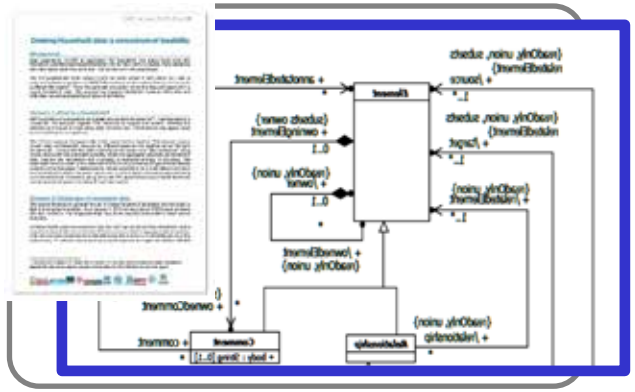
Yes / No

Semantic conformance checked *automatically* by tools built manually based on free text

Checking Semantic Conformance, More Auto

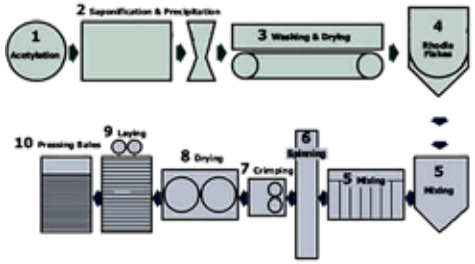
Modeling language standard

- § Graphics:
  - – Circles
  - – Lines
  - – Rectangles
- § Text
  - – Reserved
  - – Order
- § Linguistic terms:
  - – Blocks, Item Flows
  - – Activity, Interaction
- § Using terms:
  - – Make an activity the behavior for a block

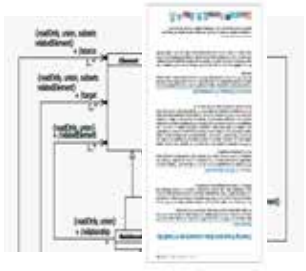


Using a Modeling Language

System model



reuses



Domain library

Interpreting a specification

Operations

Real or virtual things being modeled

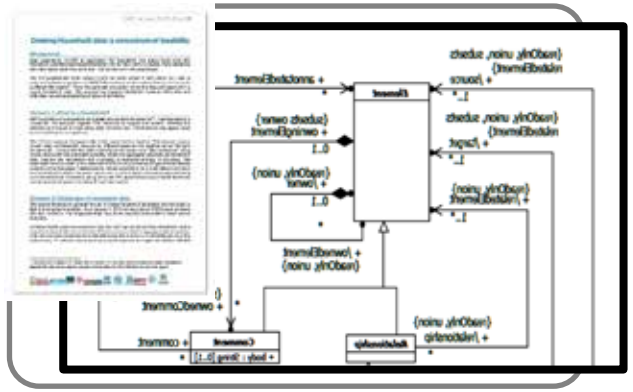
Yes / No

Semantic conformance checked automatically by tools built manually based on *formal models*

Checking Semantic Conformance, Most Auto

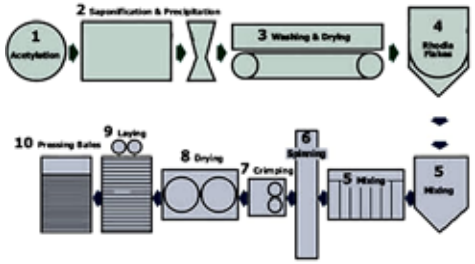
Modeling language standard

- § Graphics:
  - – Circles
  - – Lines
  - – Rectangles
- § Text
  - – Reserved
  - – Order
- § Linguistic terms:
  - – Blocks, Item Flows
  - – Activity, Interaction
- § Using terms:
  - – Make an activity the behavior for a block

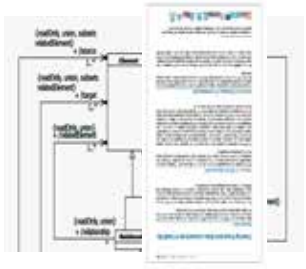


Using a Modeling Language

System model



reuses



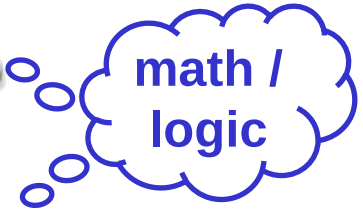
Domain library

Interpreting a specification

Operations

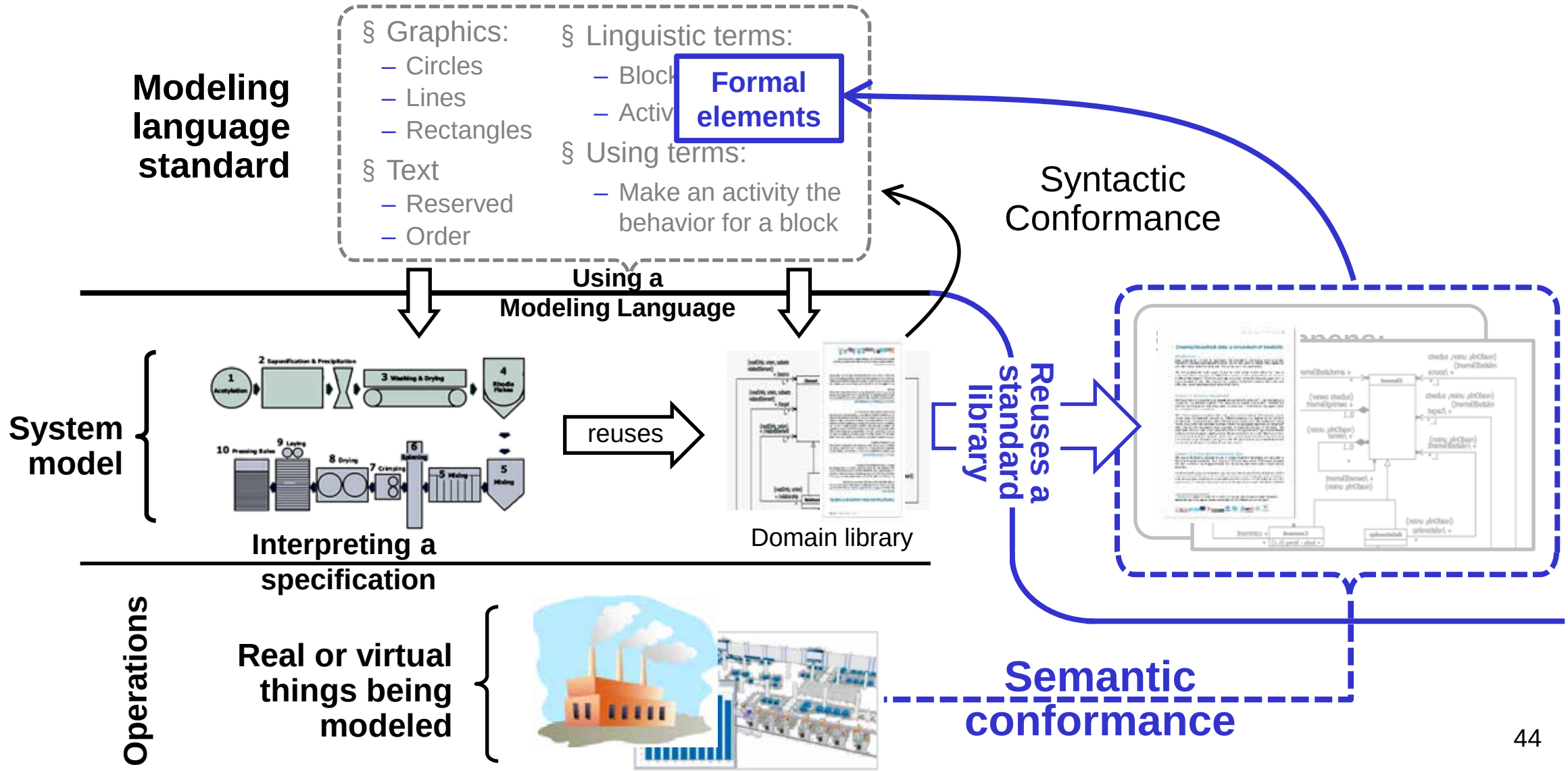
Real or virtual things being modeled

Yes / No

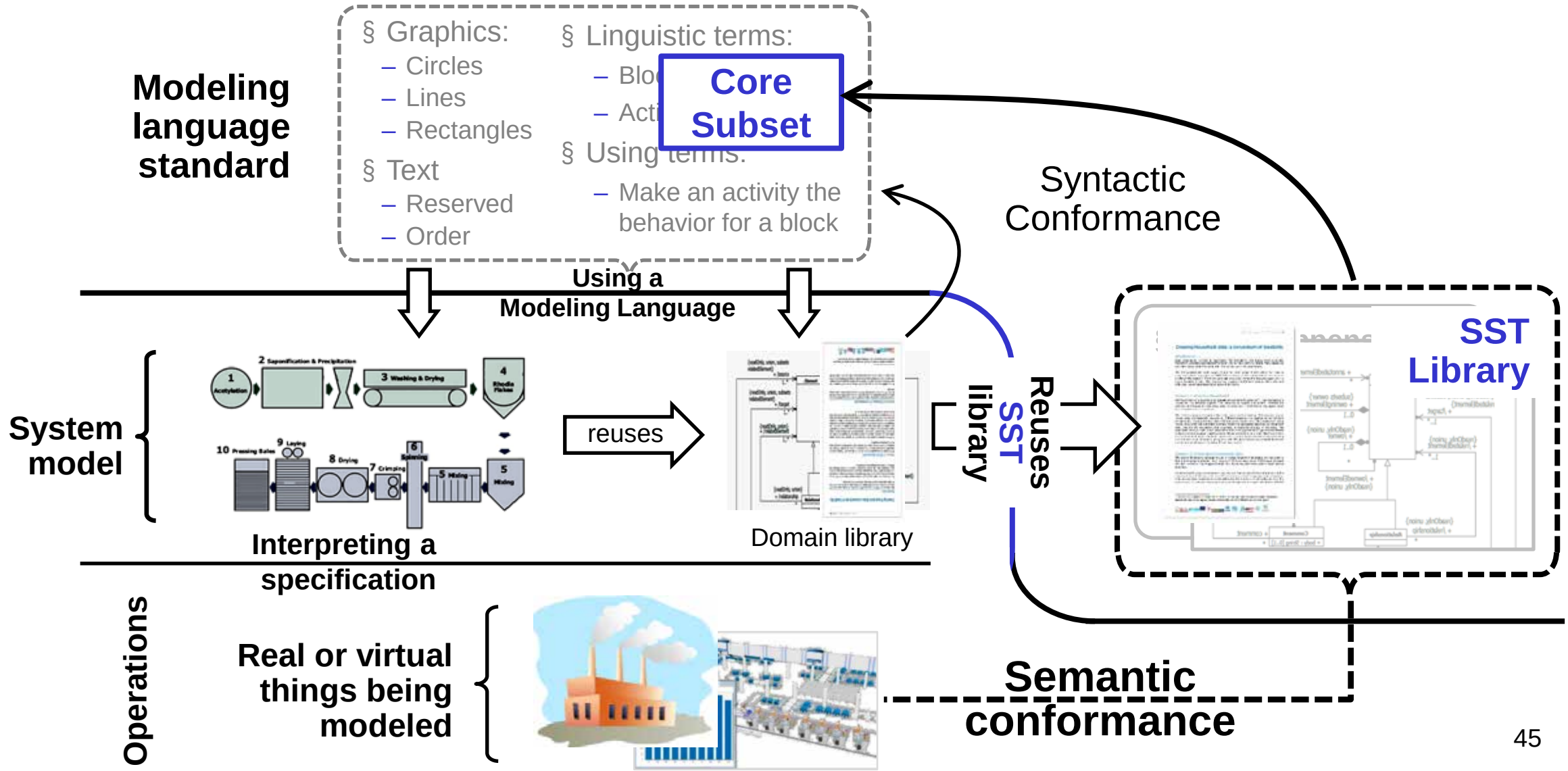


Semantic conformance checked automatically by tools built manually for checking *all formal models*

Standard Semantic Models



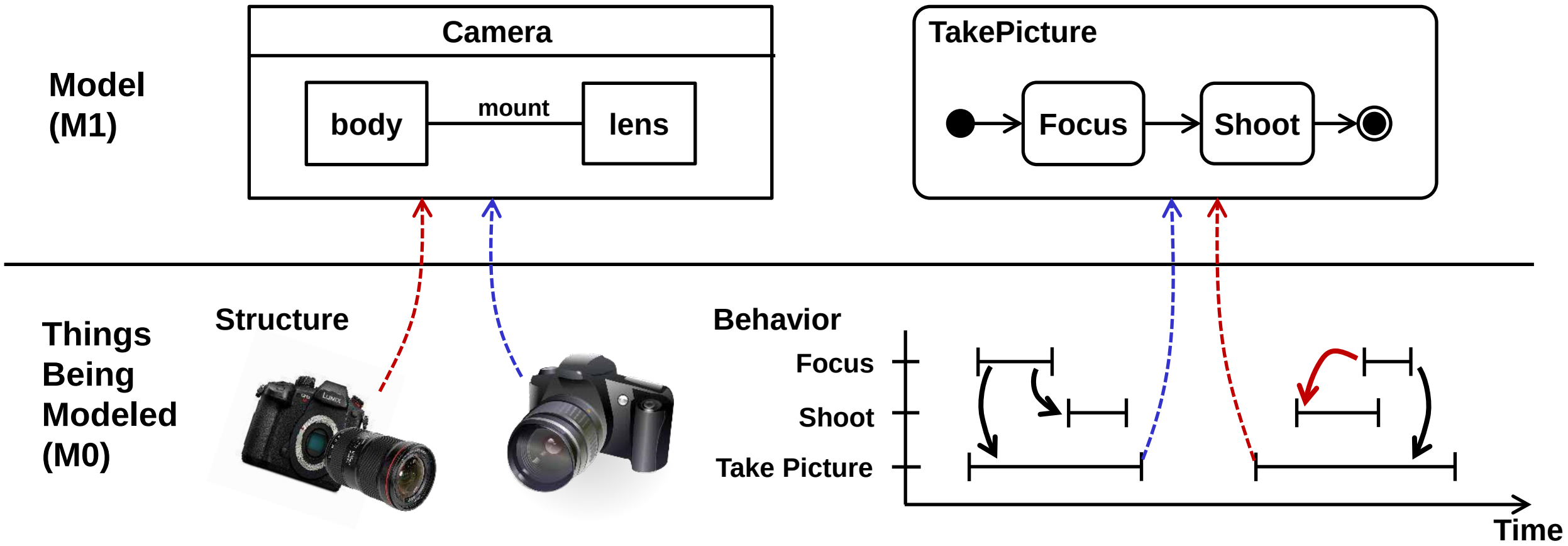
Standard Semantic Models (SST)



Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- Solution
  - The “S” Words
  - Standardizing Semantics
  - **Conformance = Classification**
  - Formalizing Semantics (ie, a little math)
  - SysML 2 Semantics
- The “O” Word
- Summary

Conformance = Classification



- Things (structural and behavioral) that do/not conform to (are/not classified by) their models

Classification Synonyms

Classified by

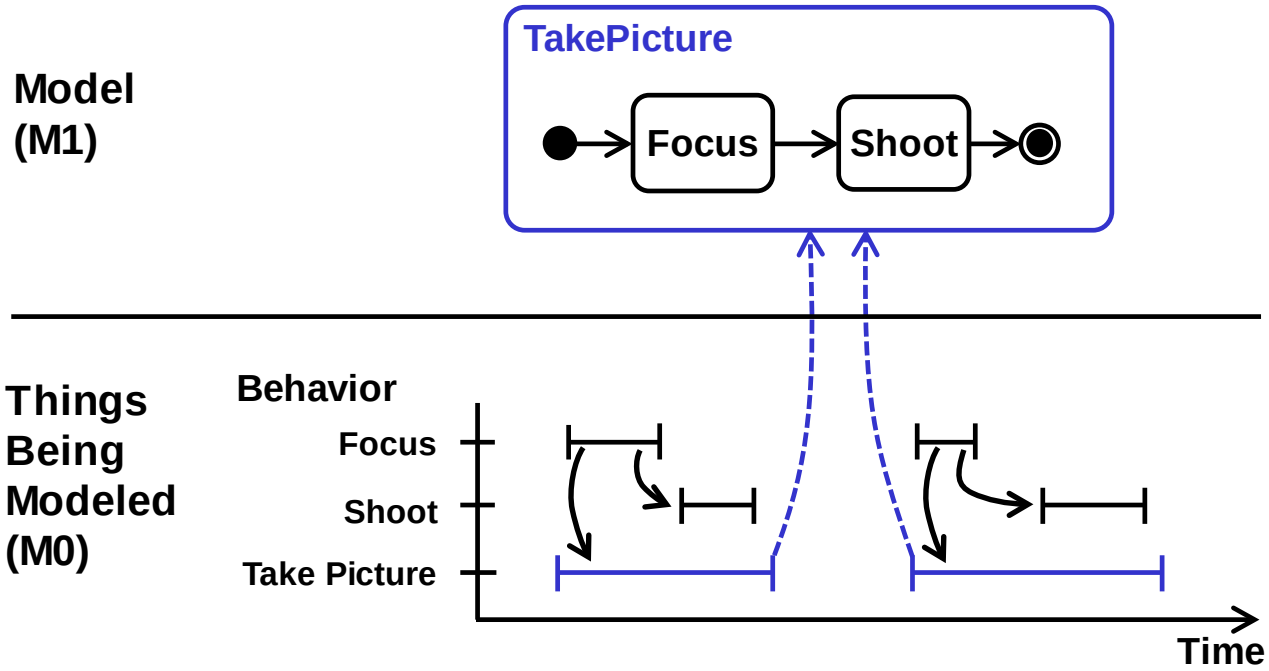
Modeled by

Specified by

Conforms to

Follows

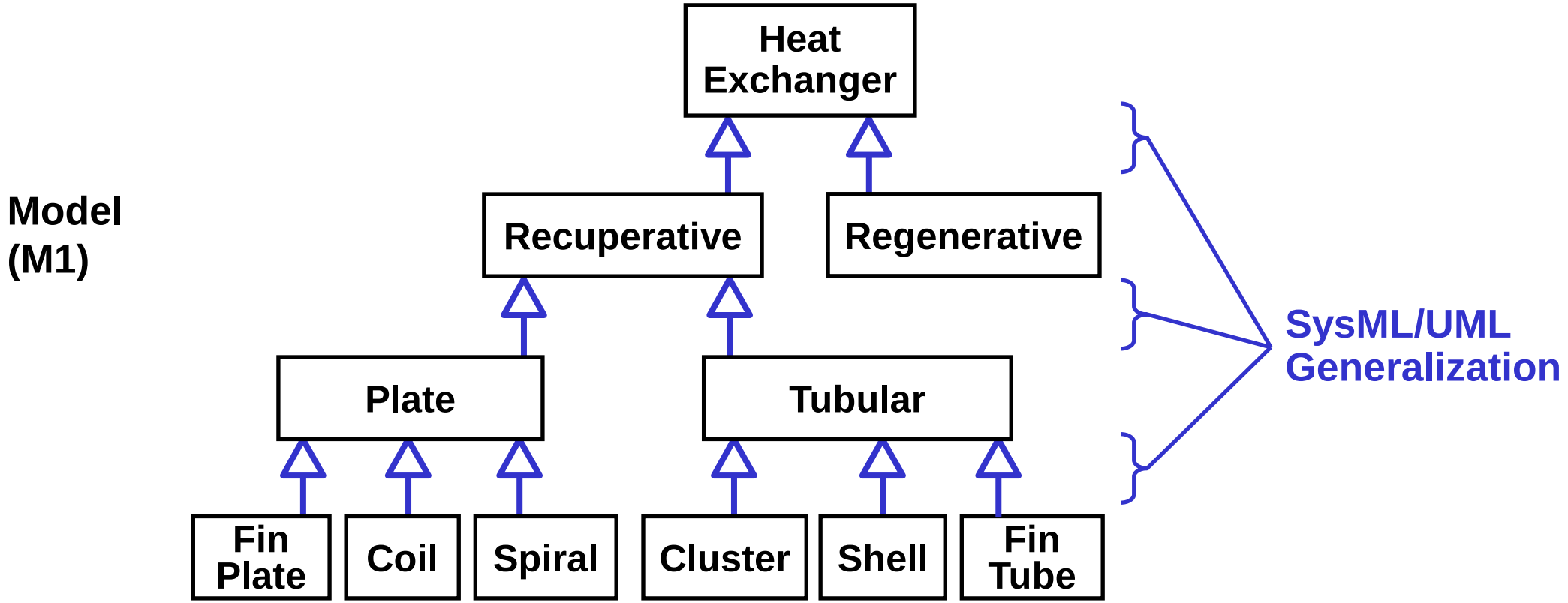
Satisfies (logically)



Not quite: "Instance of" (in the OO sense)

Not *at all* : "Execution of" (MES/software sense)

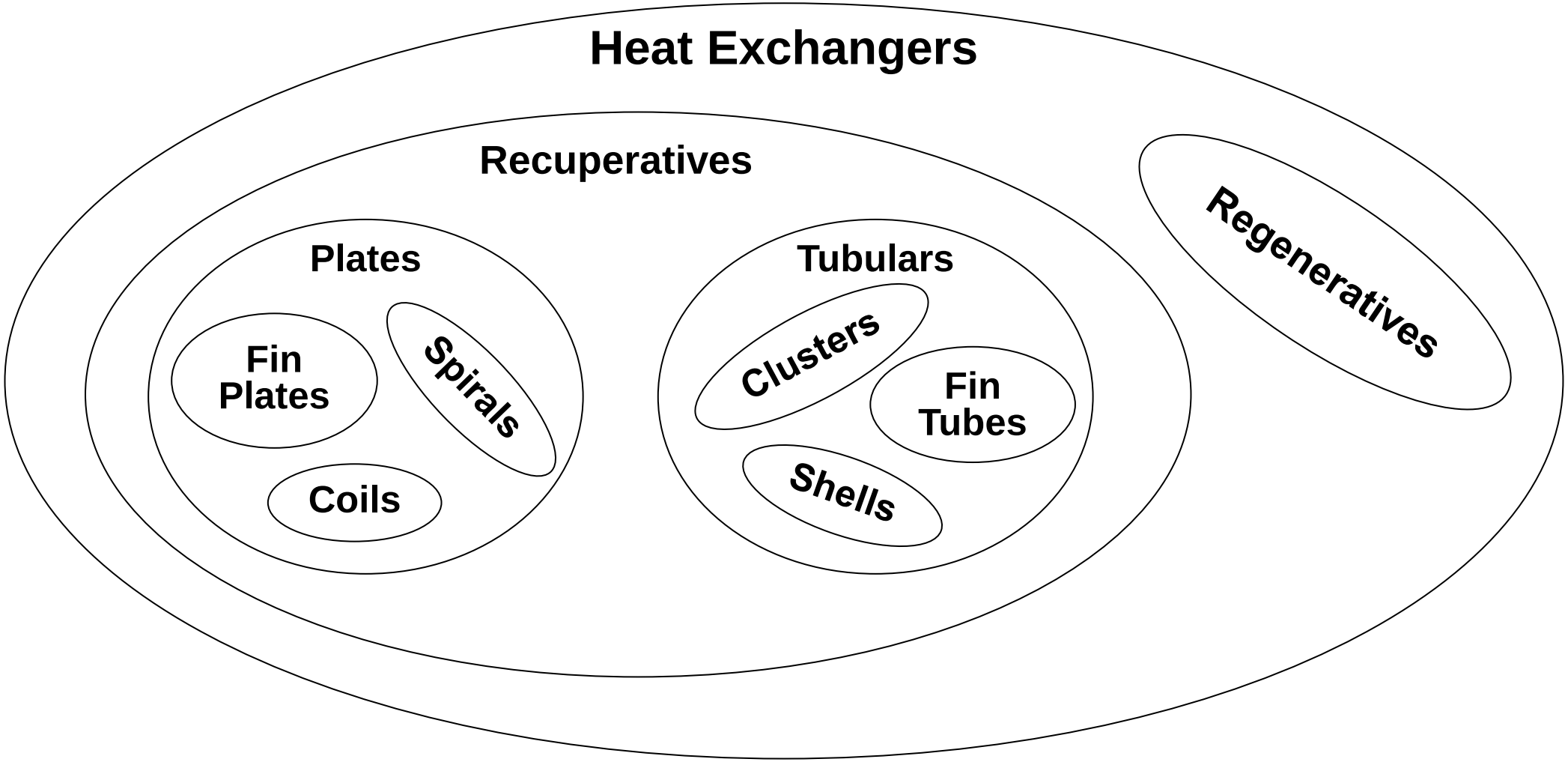
Taxonomies



- "Sub"classification …
- …of real or virtual things.

http://mechanicalgalaxy.blogspot.com/2011/05/heat-exchangers-classification-types.html

Venn Diagrams



- More accessible notation for taxonomies
  - but less scalable.

Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- Solution
  - The “S” Words
  - Standardizing Semantics
  - Conformance = Classification
  - **Formalizing Semantics (ie, a little math)**
  - SysML 2 Semantics
- The “O” Word
- Summary

Informal Semantics

UML/SysML Generalization

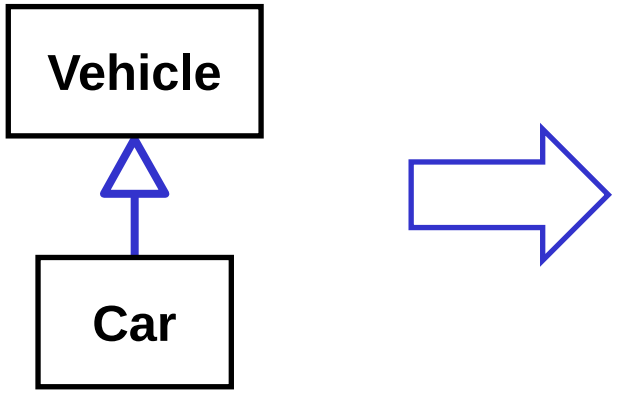
From UML 2.5 Specification:

9.9.7 Generalization [Class]

9.9.7.1 Description

A Generalization is a taxonomic relationship between a more general Classifier and a more specific Classifier. Each instance of the specific Classifier is also an instance of the general Classifier. The specific Classifier inherits the features of the more general Classifier. A Generalization is owned by the specific Classifier.

"Each instance of the specific classifier is also an instance of the general classifier"



"Every instance of Car is an instance of Vehicle"

iow

"Every Car is a Vehicle"

iow

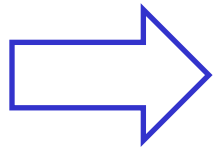
"Cars are vehicles"

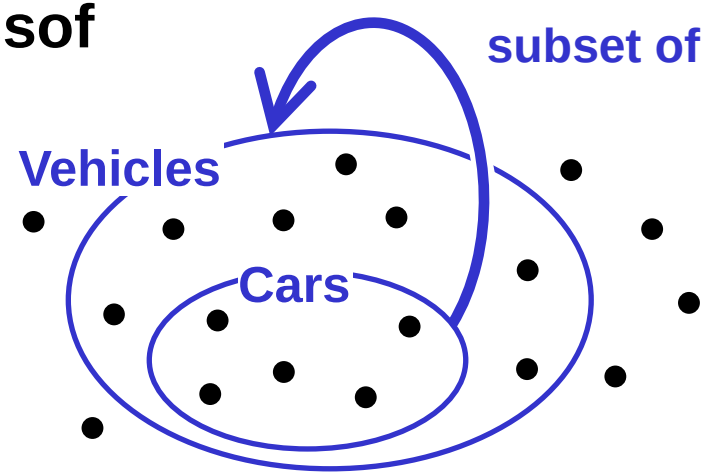
How can this be specified more precisely?

Mathematical Semantics

OWL SubClassof

SubClassOf (Car, Vehicle)





● = a single real or virtual thing

From OWL 2 Direct Semantics:

2.3 Satisfaction in an Interpre

An axiom or an ontology is *satisfie*

2.3.1 Class Expression Axioms

Satisfaction of OWL 2 class expression axioms in *I* is defined as shown in Table 5.

Table 5. Satisfaction of Class Expression Axioms in an Interpretation

Axiom	Condition
SubClassOf(CE_1 CE_2)	$(CE_1)^C \subseteq (CE_2)^C$

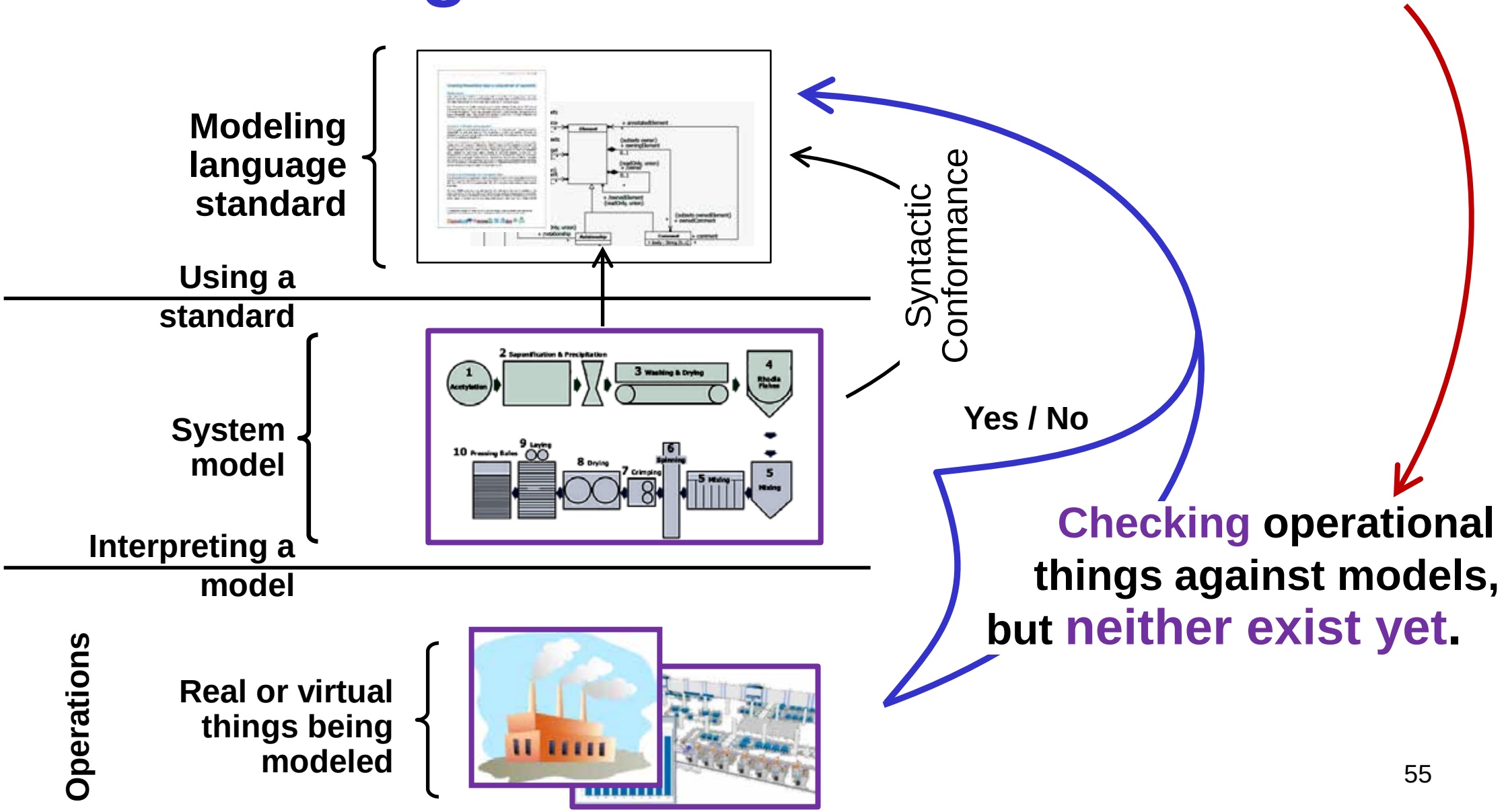
Axiom	Condition
SubClassOf(CE_1 CE_2)	$(CE_1)^C \subseteq (CE_2)^C$

OWL 2 Direct Semantics https://www.w3.org/TR/owl2-direct-semantics/

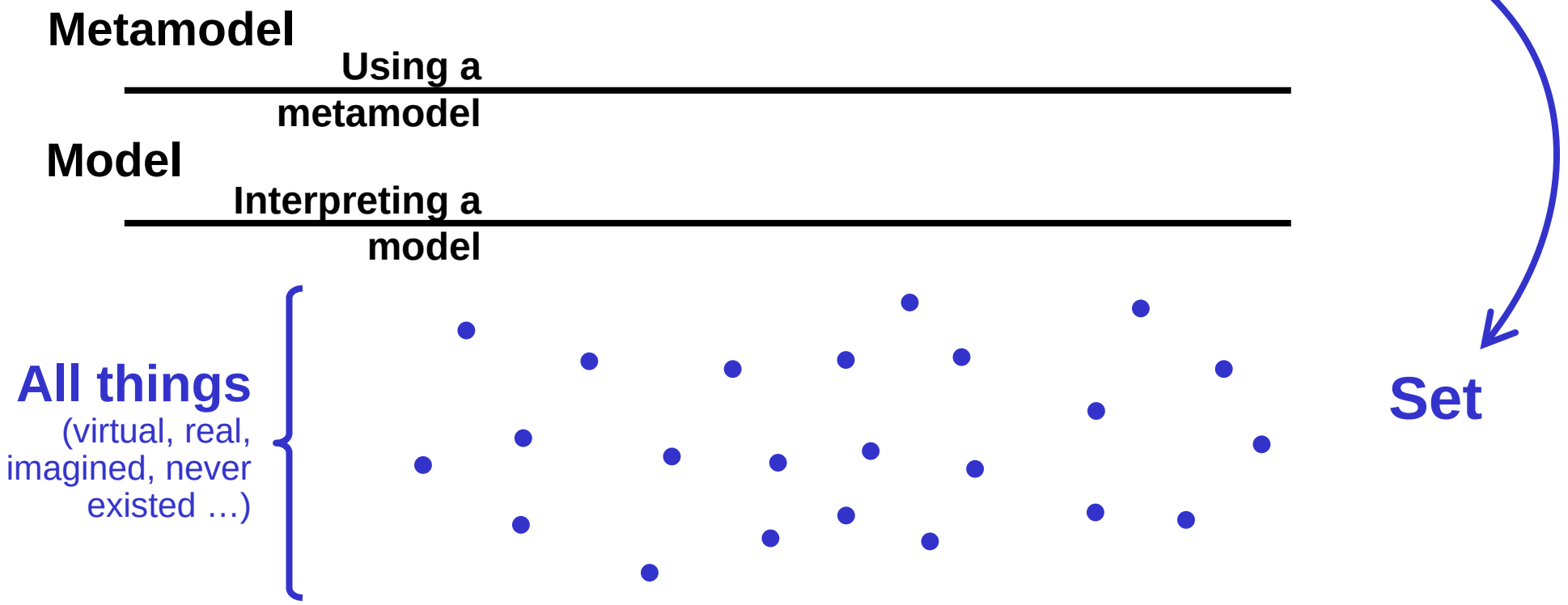
SST and OWL

- **Most of SST semantics is equivalent to OWL.**
  - **Emulated its style and notation.**
  - **Exceptions covered in the next section.**
- **This section covers SST as it overlaps OWL**
  - **uses some SST terms …**
  - **… with OWL semantics and notation conventions.**
- **Next section updates semantics to SST.**

Standardizing Semantic Conformance?



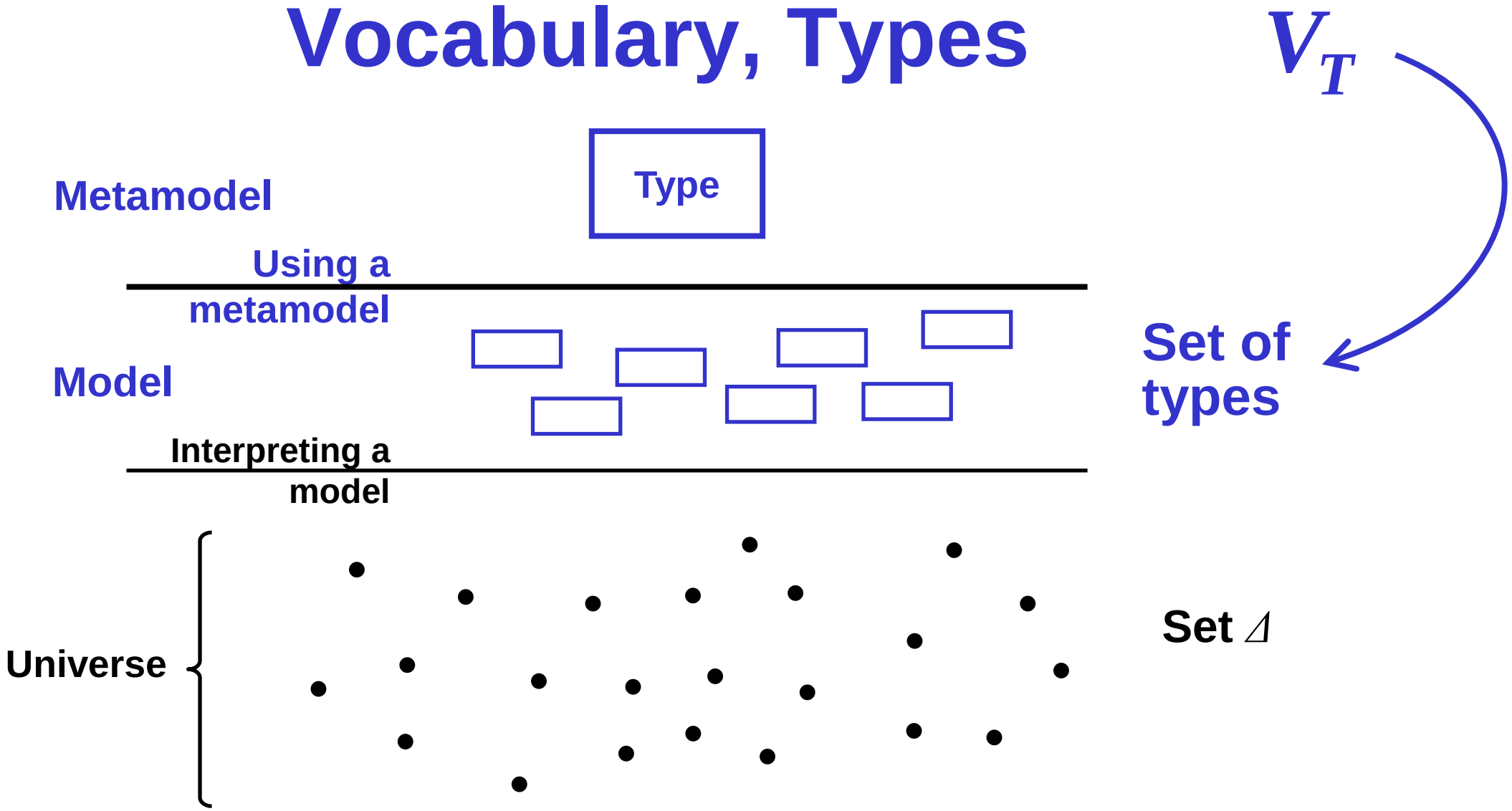
Universe Δ



- **Everything, anything, no restrictions, don't know anything about them, how many, etc.**
- **For interpreting models.**

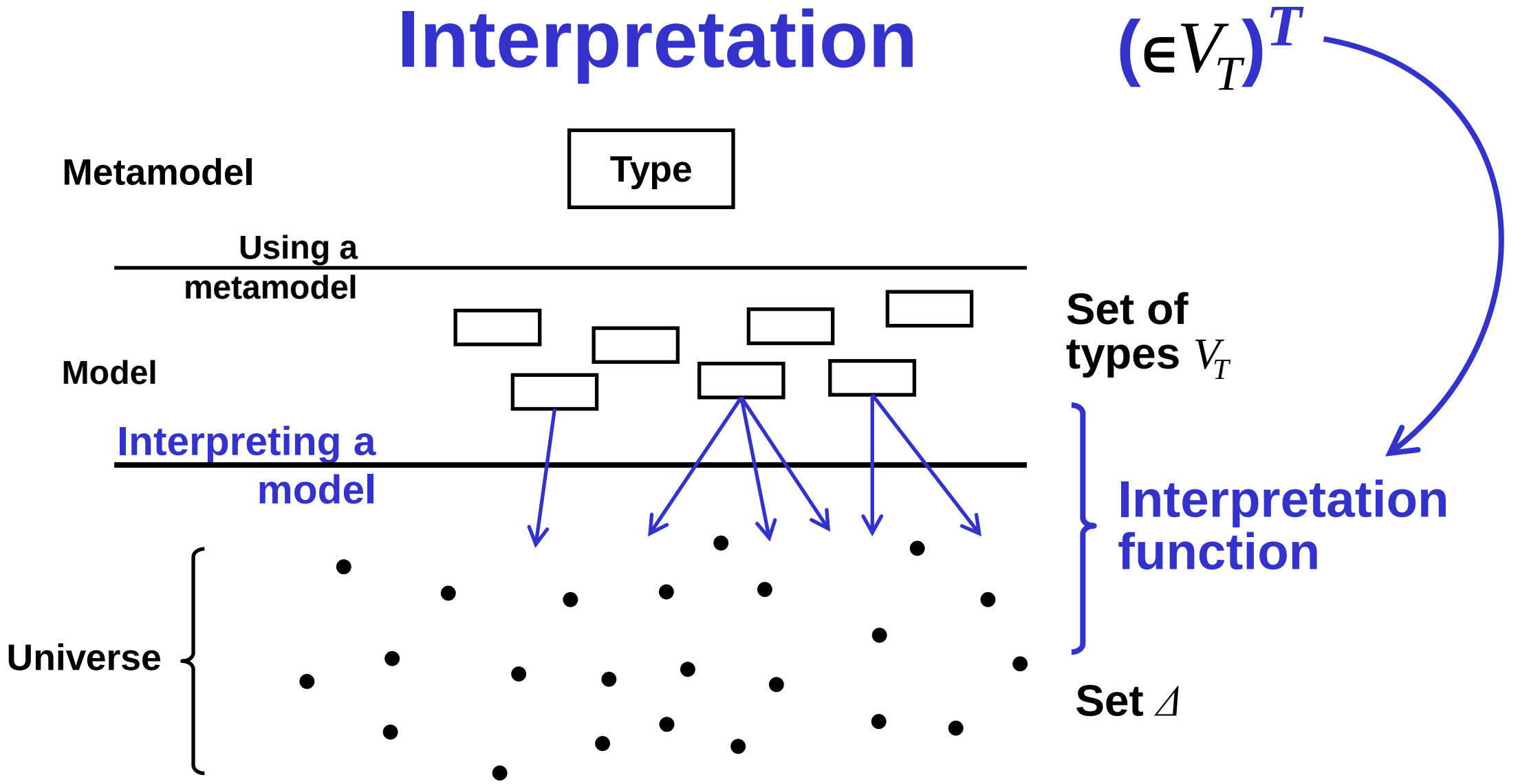
OWL 2 Direct Semantics https://www.w3.org/TR/owl2-direct-semantics/

Vocabulary, Types



- Beginning of syntax for model elements.
  - At least for elements affecting real & virtual things.

Interpretation



- Links model elements to (mathematical structures made up of) things in the universe.

SysML 2 Math Terms

7.3.1.2 Mathematical Preliminaries

The following are model theoretic terms, explained in terms of this specification:

- *Vocabulary*: Model elements conforming to abstract syntax and additional restrictions given in this subclause.
- *Universe*: All (real or virtual) things the vocabulary could possibly be about.
- *Interpretation*: The relationship between vocabulary and mathematical structures made of elements of the universe.

Mini-Glossary

The *semantics* of KerML are restrictions on the interpretation relationship, given in this subclause and the Semantics subclauses. This subclause also defines the above terms for KerML. They are used by the mathematical semantics in the rest of the specification.

A vocabulary $V = (V_T, V_C, V_F)$ is a 3-tuple where:

- V_T is a set of types (model elements classified by Type, see 7.3.2.3).
- $V_C \subseteq V_T$ is a set of classifiers (model elements classified by Classifier, see 7.3.3.3), including at least *Base::Anything* from KerML model library, see 8.2).
- $V_F \subseteq V_T$ is a set of features (model elements classified by Feature, see 7.3.4.3), including at least `Base::things` from the KerML model library (see 8.2).
- $V_T = V_C \cup V_F$

Vocabulary

An interpretation $I = (\Delta, \cdot^T)$ for V is a 2-tuple where:

- Δ is a non-empty set (*universe*), and
- $\cdot^T$ is an (*interpretation*) function relating elements of the vocabulary to sets of sequences of elements of the universe. It has domain V_T and co-domain that is the power set of S, where

  $S = \cup_{i \in Z^+} \Delta^i$

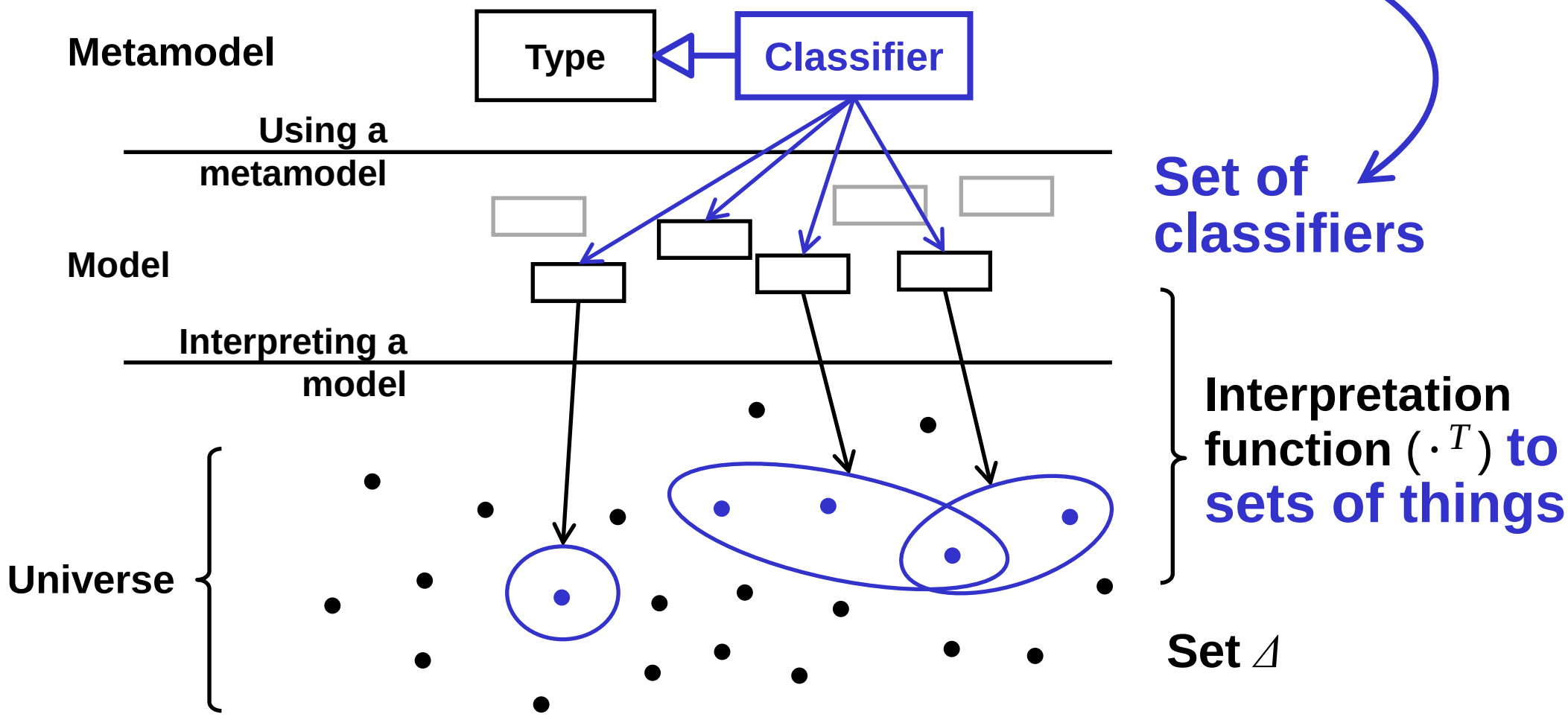
  S is the set of all n-ary Cartesian products of Δ with itself, including 1-products, but not 0-products, which are called *sequences*. The Semantics subclauses give other restrictions on the interpretation function.

The phrase *result of interpreting* a model (vocabulary) element refers to sequences paired with the element by $\cdot^T$. This specification also refers to this as the *interpretation* of the model element, for short.

Interpretation

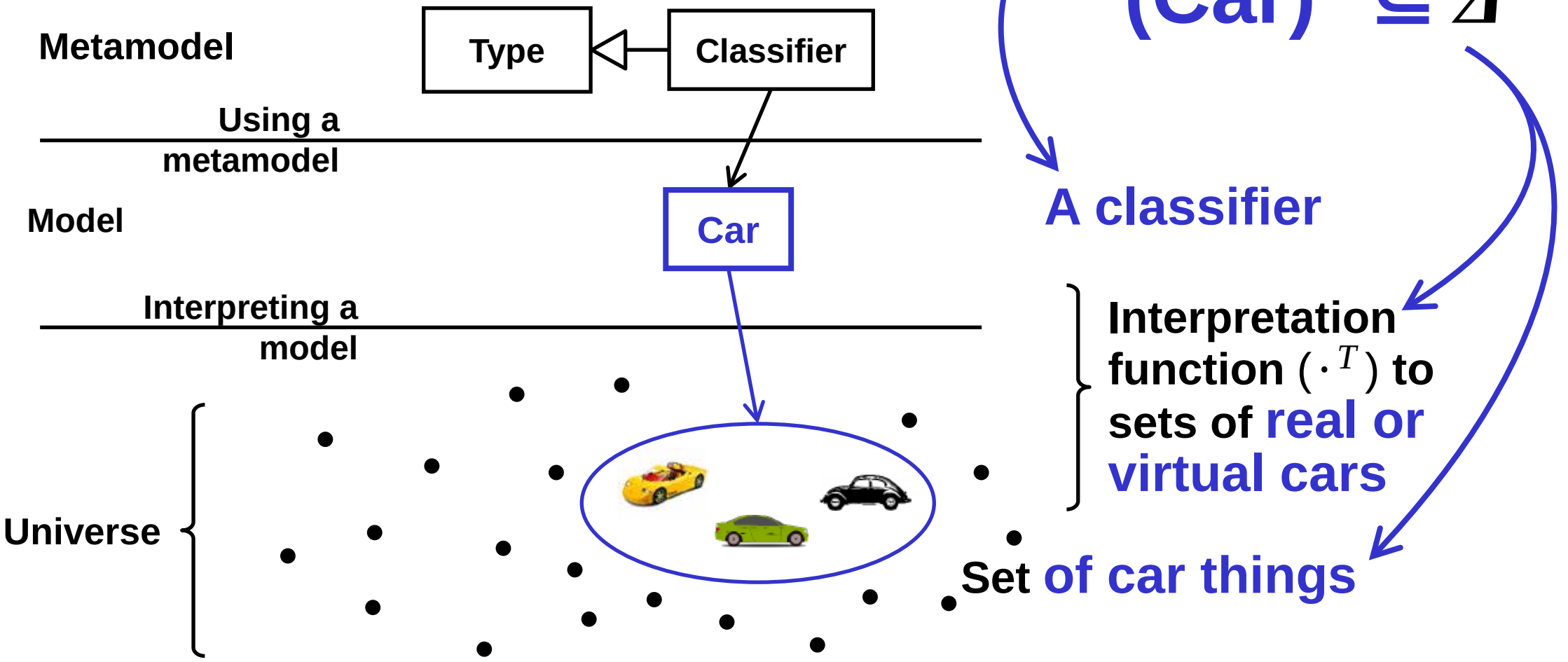
Interpretation, Classifiers

$V_C \subseteq V_T$



- **Classifiers are interpreted as sets of things in the universe.**
  - (the sets are not in the universe)

Interpretation, Classifiers, Example

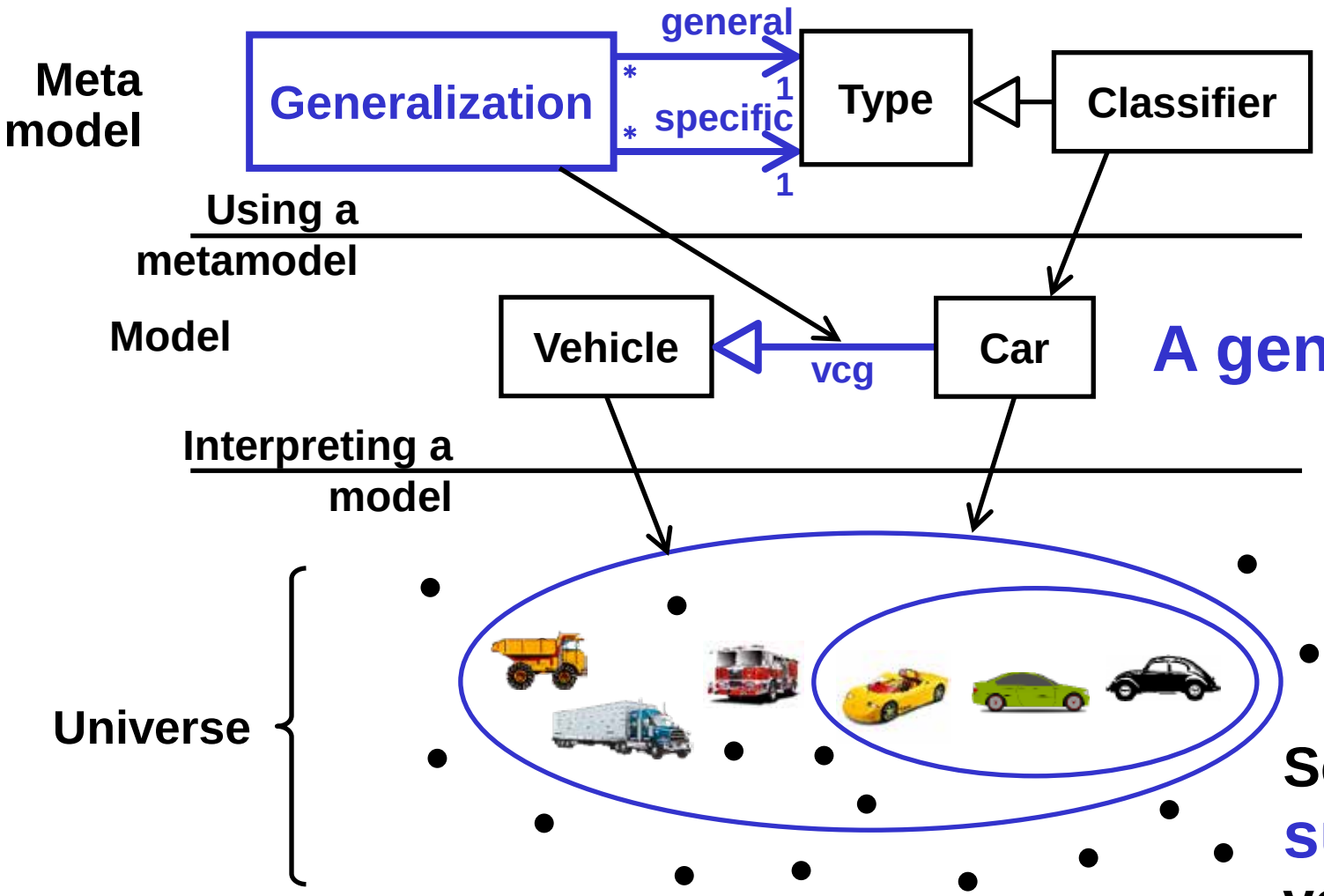


§ Car is interpreted as a set of real or virtual things.

Interpretation, Generalization

Classifier

vcg.general = Vehicle
vcg.specific = Car



$(Car)^T \subseteq (Vehicle)^T$

A generalization

Interpretation function $(\cdot^T)$ to sets of real or virtual vehicles, including cars

Set of car things is a subset of the set of vehicle things

§ Car's interpretation is a subset of Vehicle's.

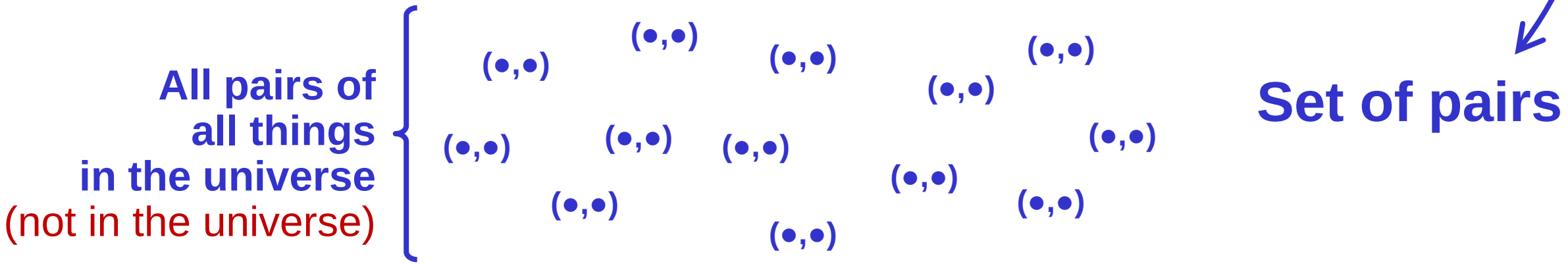
Pairs of (Things in the Universe) $\Delta \times \Delta$

Metamodel

Using a metamodel

Model

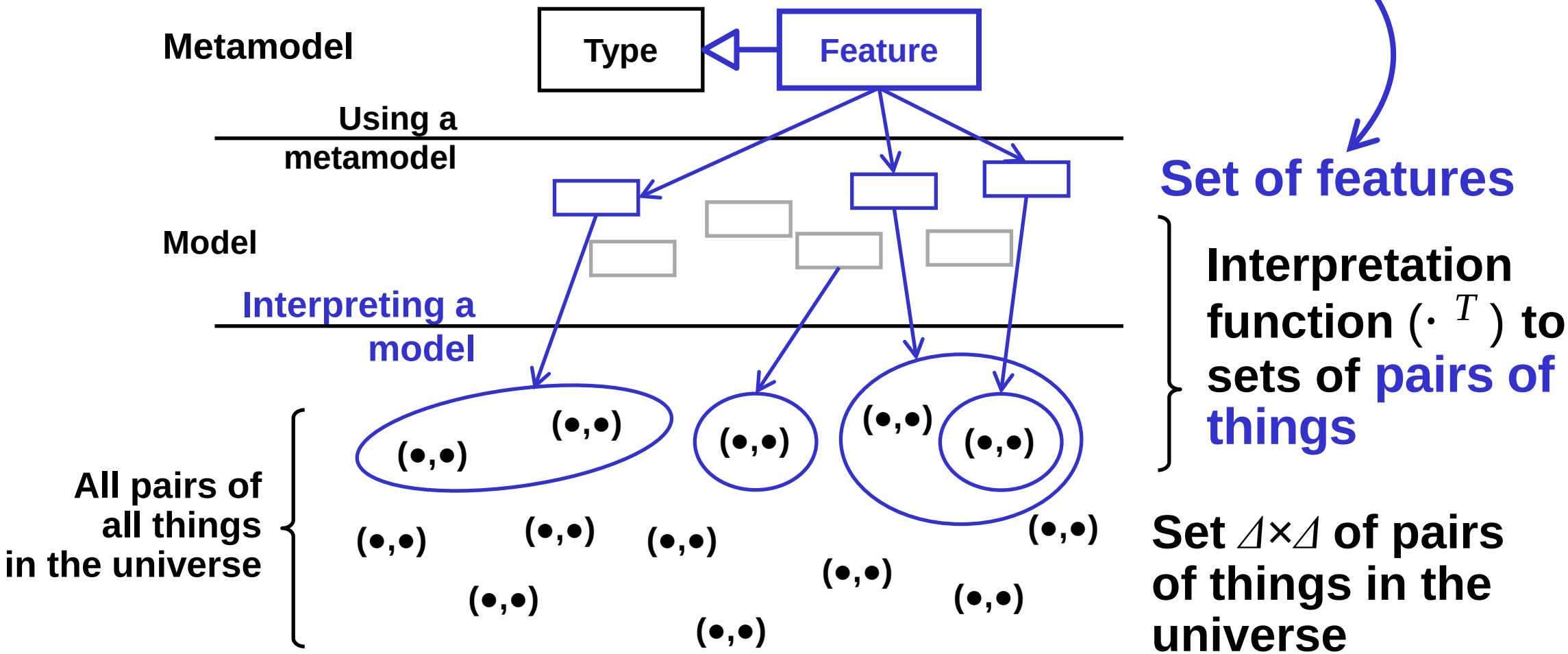
Interpreting a model



- **Every pair of anything, no restrictions on pairing, don't know anything about the pairings, etc.**
- **For interpreting relationships between things.**

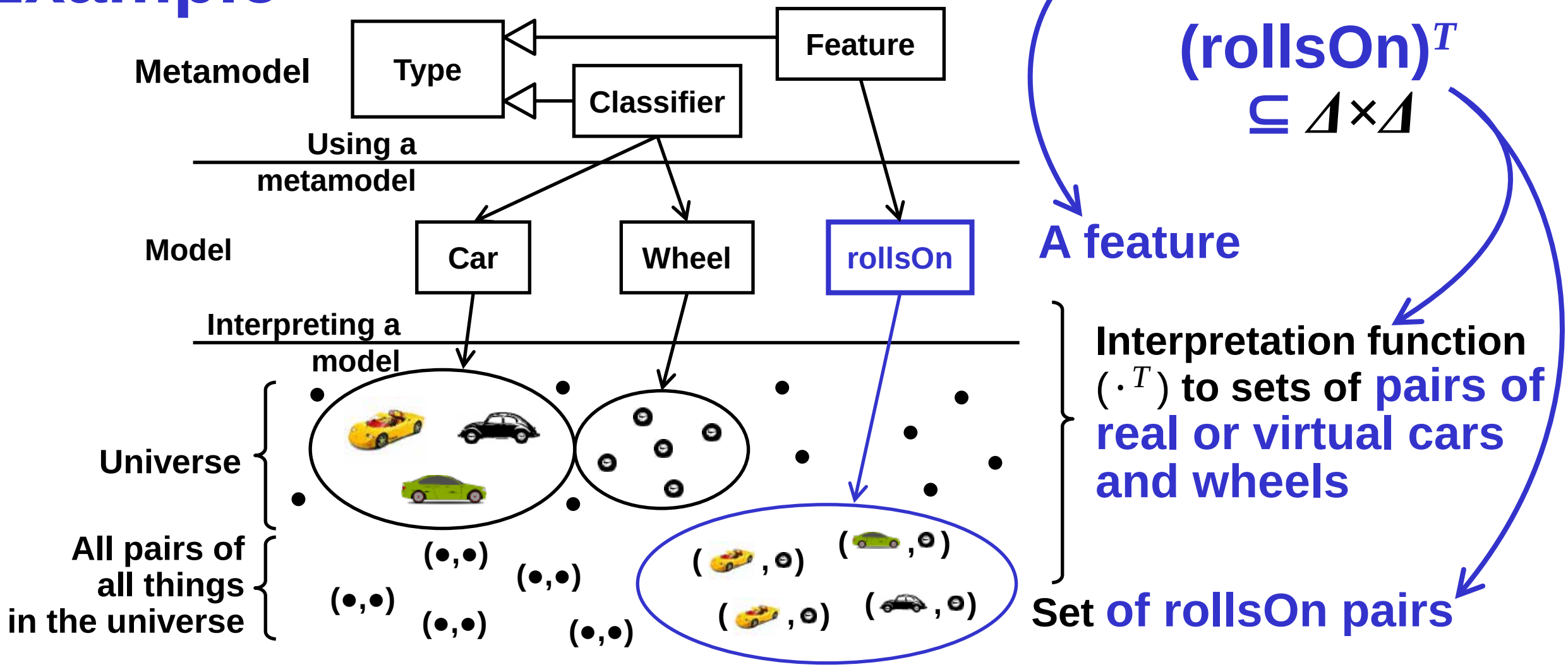
Interpretation, Features

$V_F \subseteq V_T$



- Features are interpreted as sets of pairs of things in the universe.

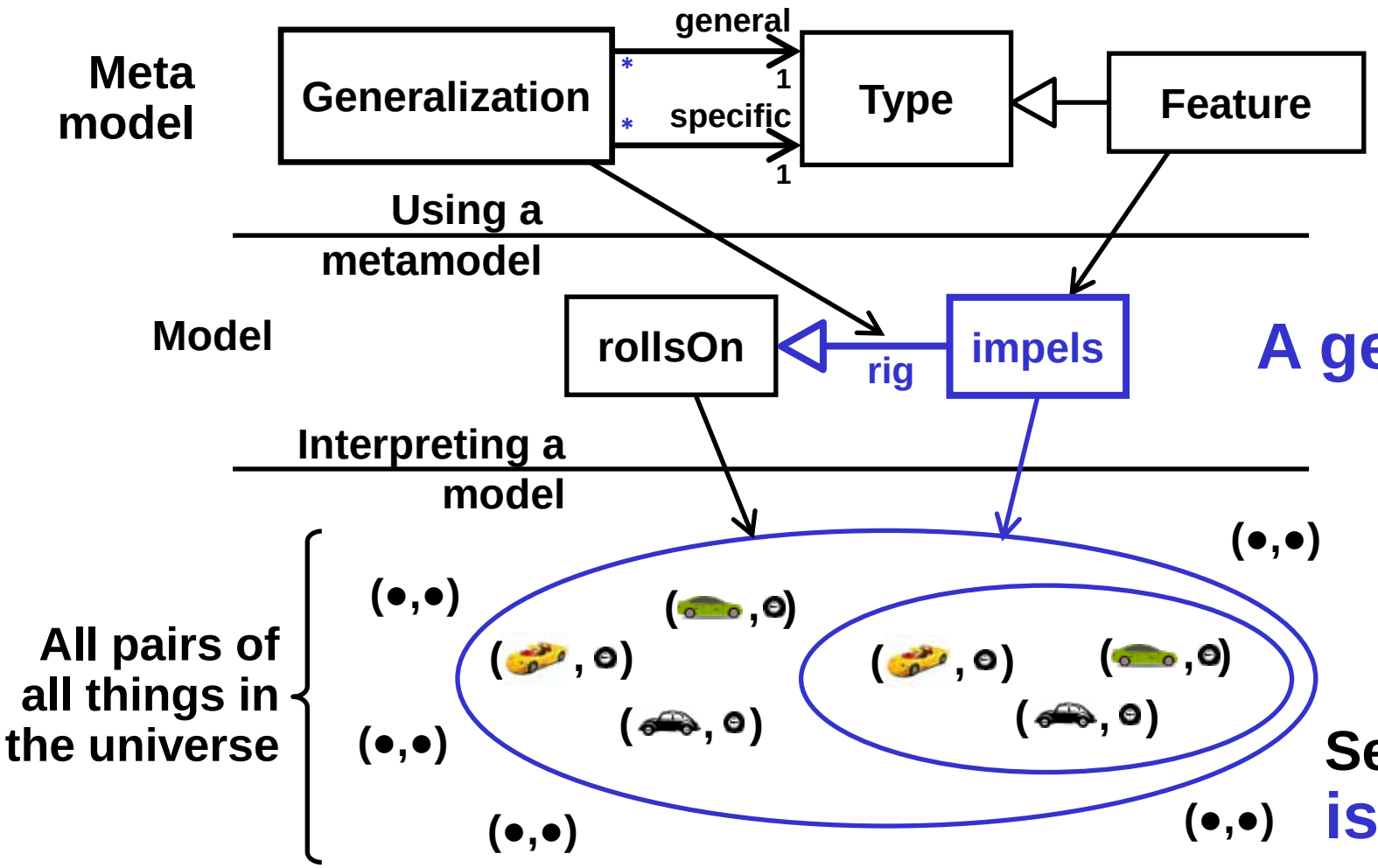
Interpretation, Features, Example



- rollsOn is interpreted as sets of pairs of real or virtual cars and wheels.

Interpretation, Generalization

Feature



rig.general = rollsOn
rig.specific = impels

$(impels)^T \subseteq (rollsOn)^T$

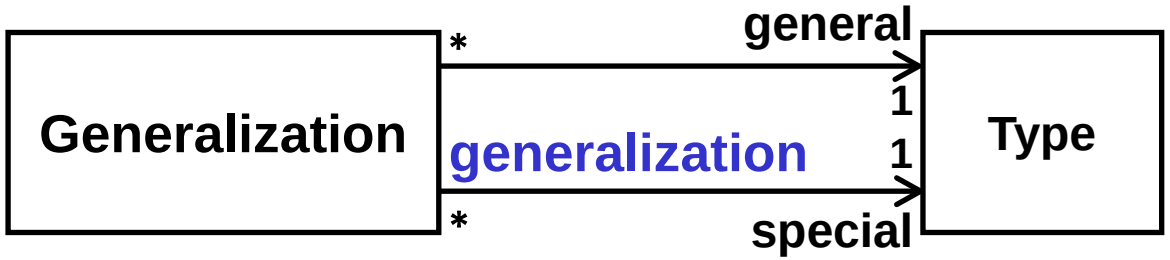
A generalization

Interpretation function $(\cdot^T)$ to pairs of real or virtual cars and wheels

Set of impels pairs is a subset of the set of rollsOn pairs

§ impel's interpretation is a subset of rollsOn's.

Generalization Math



Statement that must be true

Variables — **∀: For all possible values of the variables**

$$\forall\ t_g, t_s \in V_T \qquad t_g \in t_s.\text{generalization.general} \Rightarrow (t_s)^T \subseteq (t_g)^T$$

Variable values must be types (from a model)

general type — *special type*

if — **then**

how model constrains real or virtual things

7.3.2 Types 7.3.2.4 Semantics

1. All sequences in the interpretation of a Type are in the interpretations of its generalizing Types.

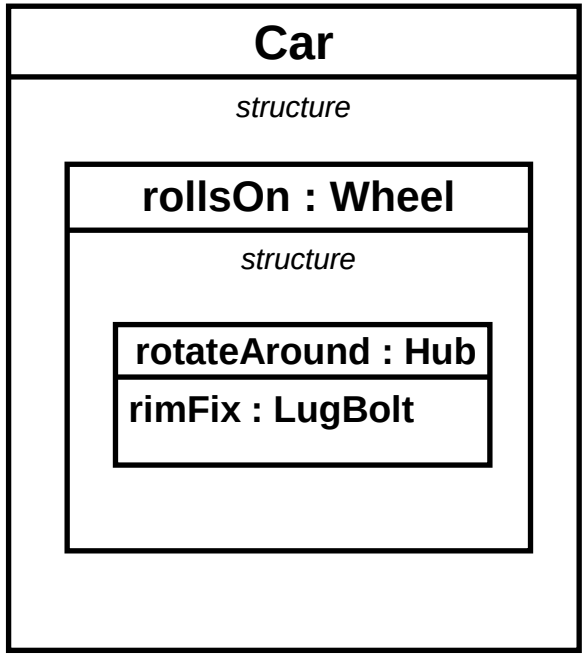
§ **Generalization = subsetting interpretations**

Overview

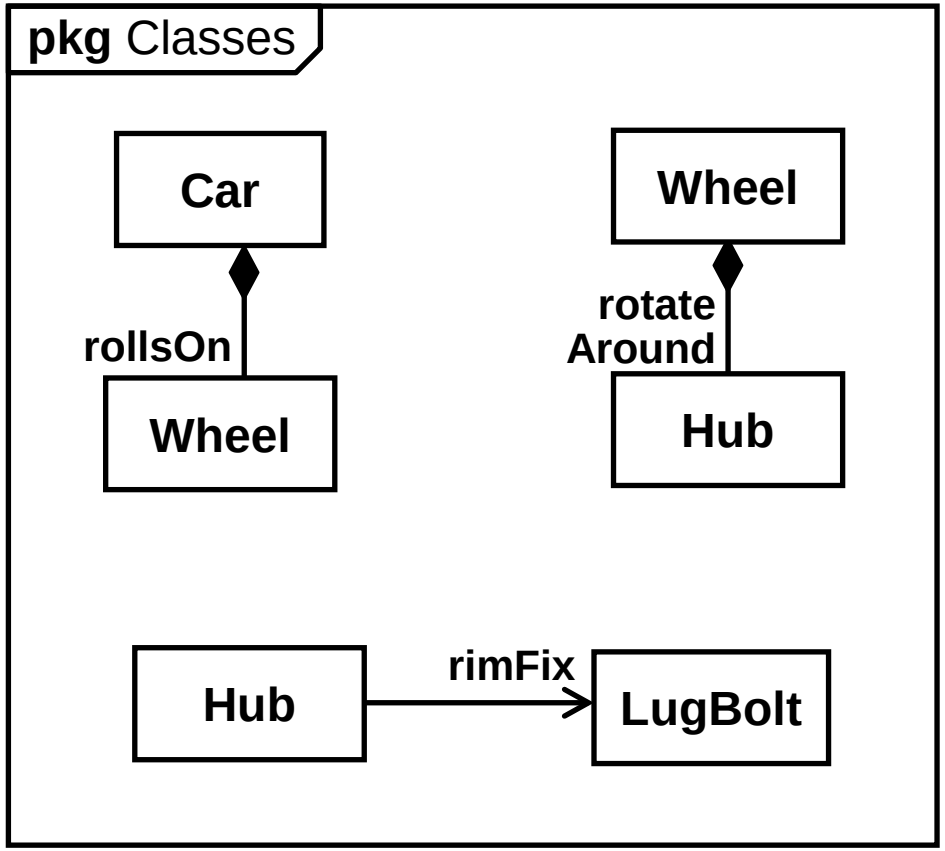
- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- Solution
  - The “S” Words
  - Standardizing Semantics
  - Conformance = Classification
  - Formalizing Semantics (ie, a little math)
  - **SysML 2 Semantics**
- The “O” Word
- Summary

Visual Nesting ≠ Class/Property Modeling

Model

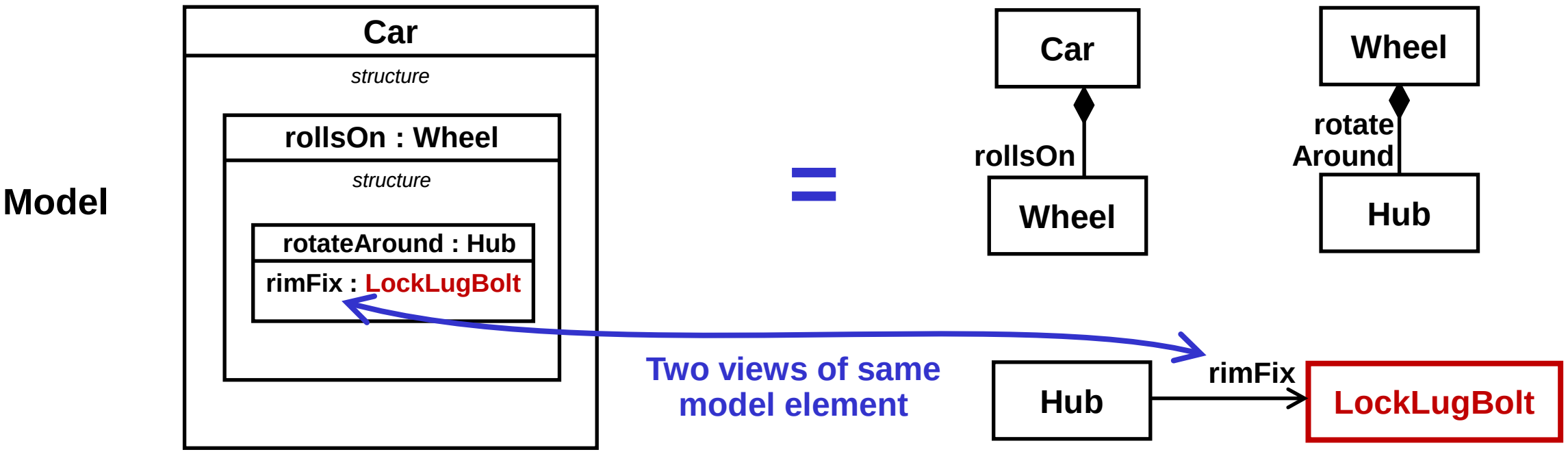


=



- **Structure diagrams same as class diagramns**
  - **as far as visual nesting goes.**

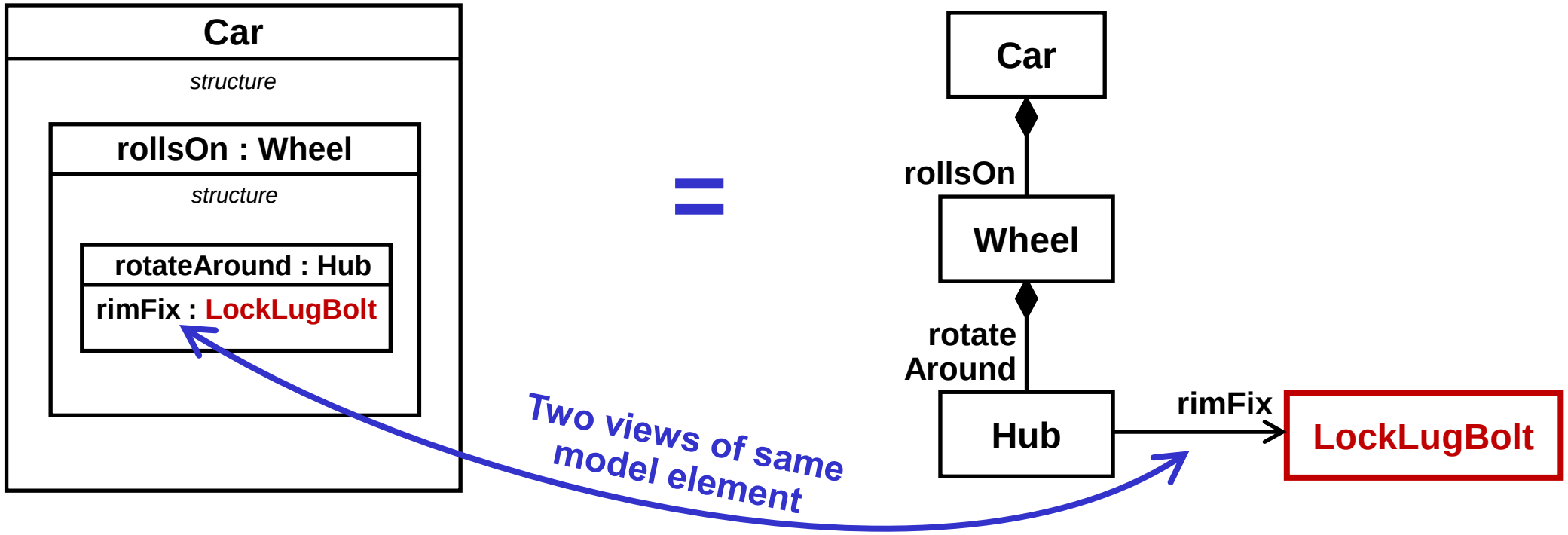
Visual Nesting ≠ Class/Property Modeling



- Don't want
  - All hubs to use lock lugbolts.
  - All wheels to have hubs with lock lugbolts.
- Just the hubs in wheels that are in cars.

Visual Nesting ≠ Class/Property Nesting

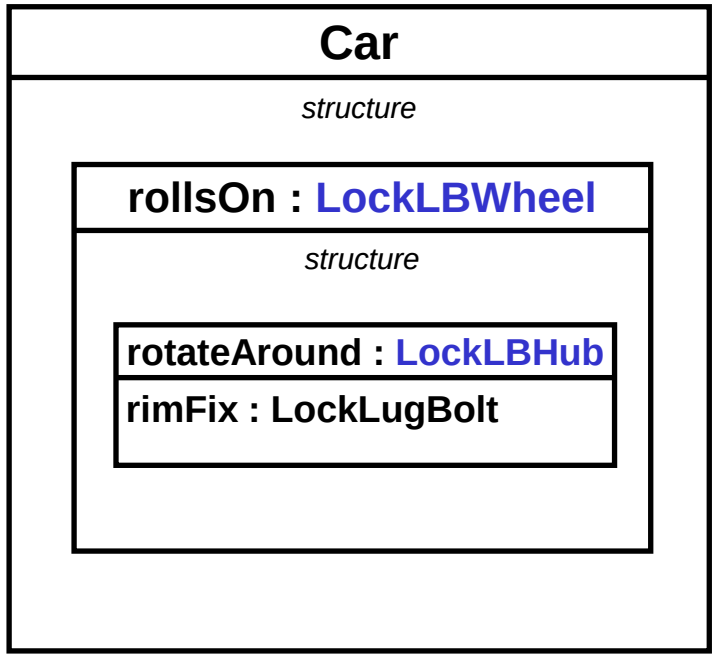
Model



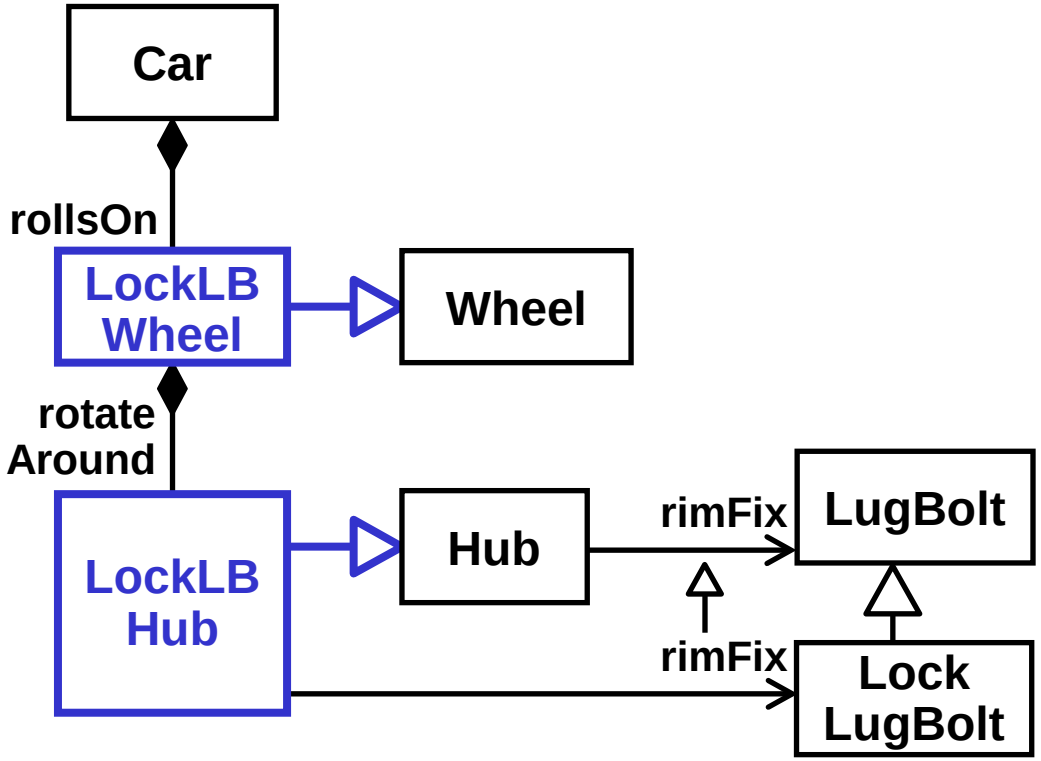
- Doesn't matter how class diagrams are drawn

Visual Nesting ≠ Class/Property Modeling

Model



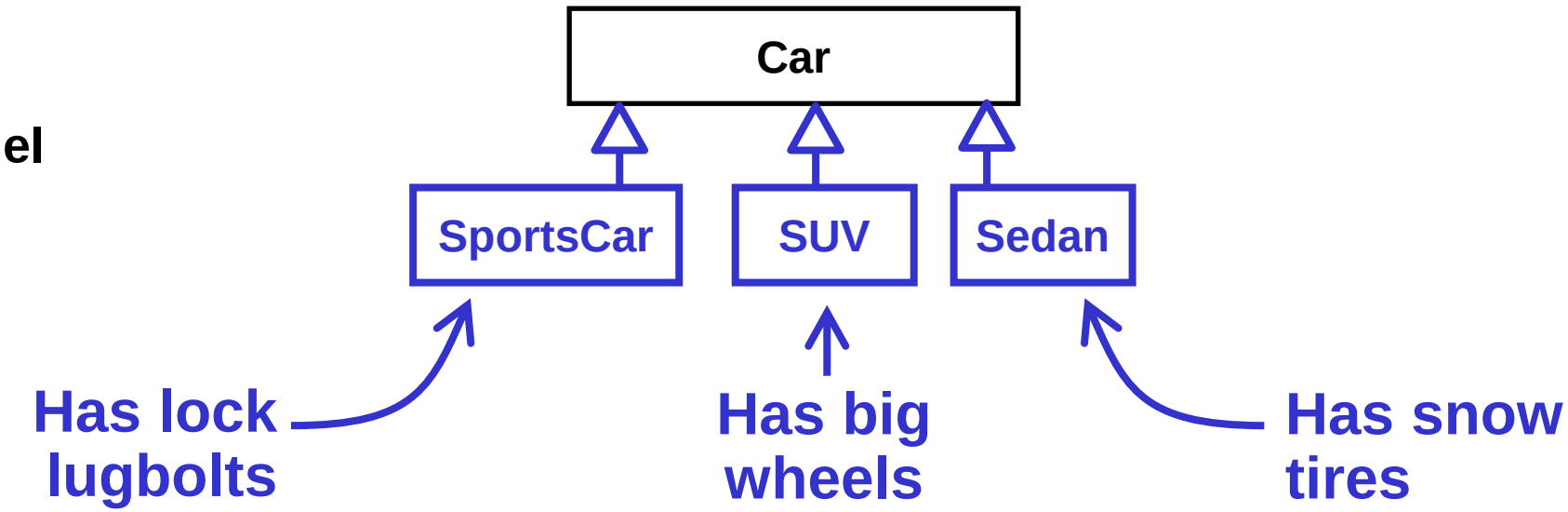
=



- Need new specialized classes all the way down the chain of properties.

Variation Modeling

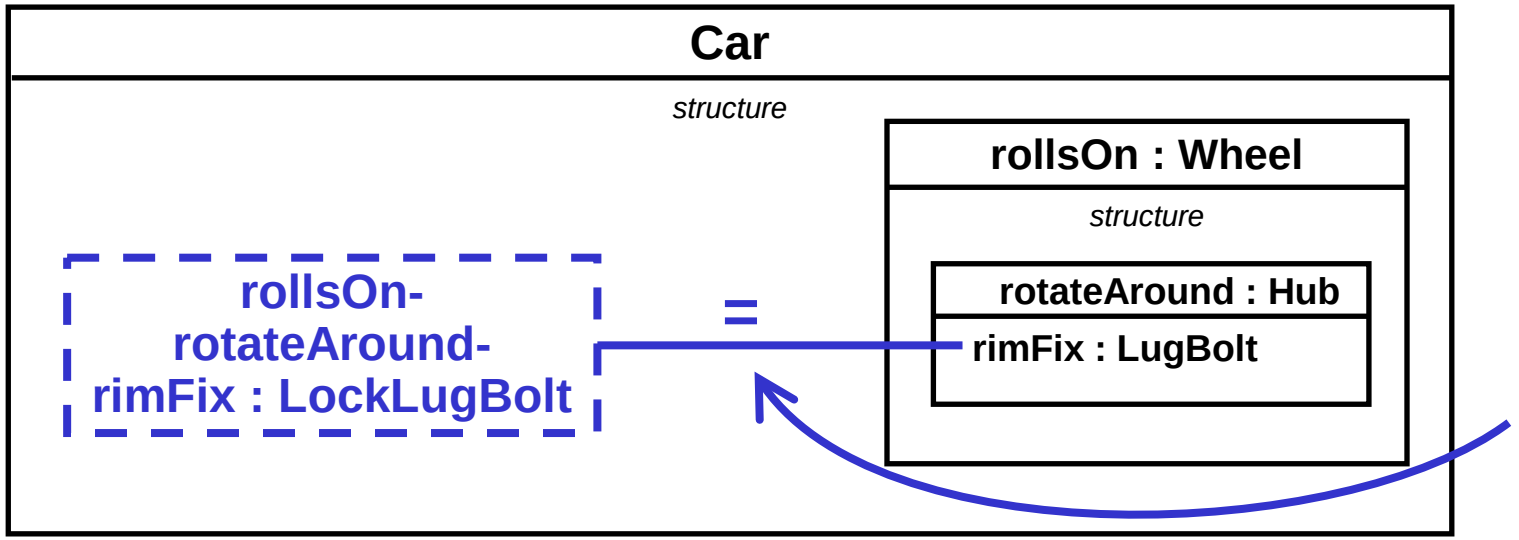
Model



§ **Need classes all the way down for all variants.**

SysML 1.4 Variant WG Archive http://www.omg.org/members/sysml-rtf-wiki/doku.php?id=rtf4:groups:variant:variants_modeling

SysML 1.x Bound References

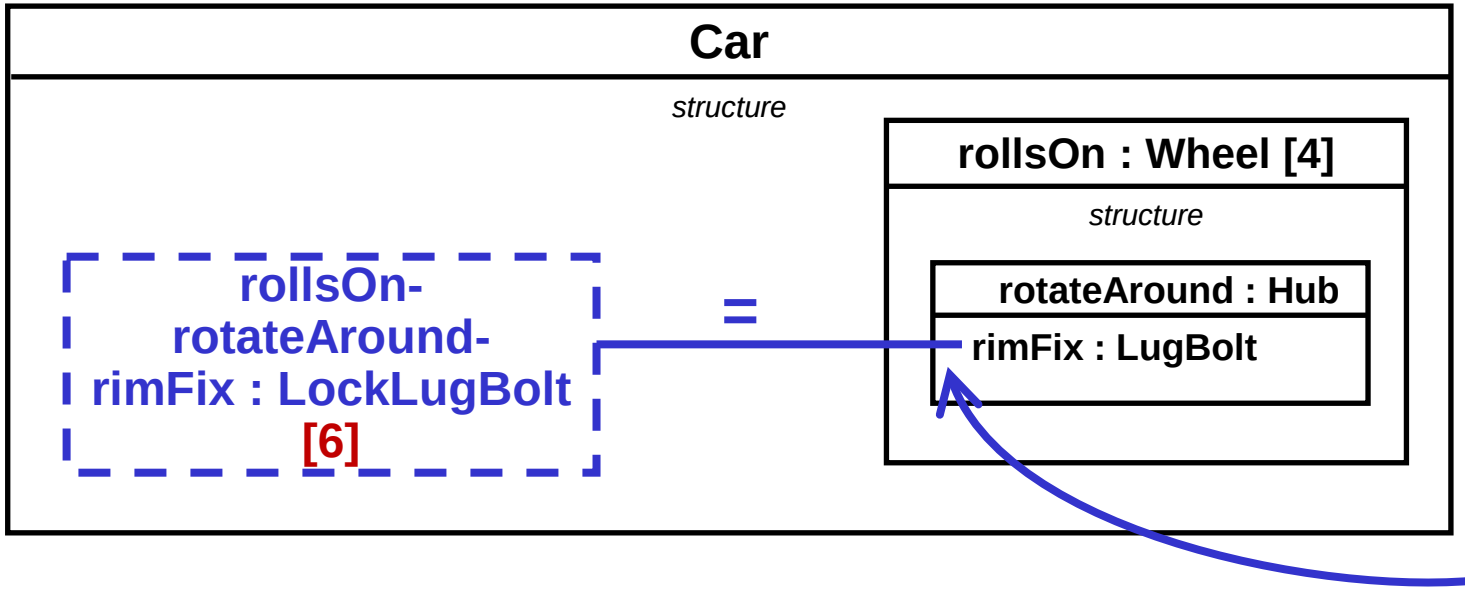


Binding means end property values are the same.

Restrictions on one apply to the other.

- **Bind new top-level property to nested one.**
  - **Restrict top-level property**
- **Pro: No new classes needed.**
- **Cons:**
  - **Restrictions on nested elements are at top-level.**
  - **Multiplicity restrictions count over all nested values.**

SysML 1.x Property Paths, Multiplicity

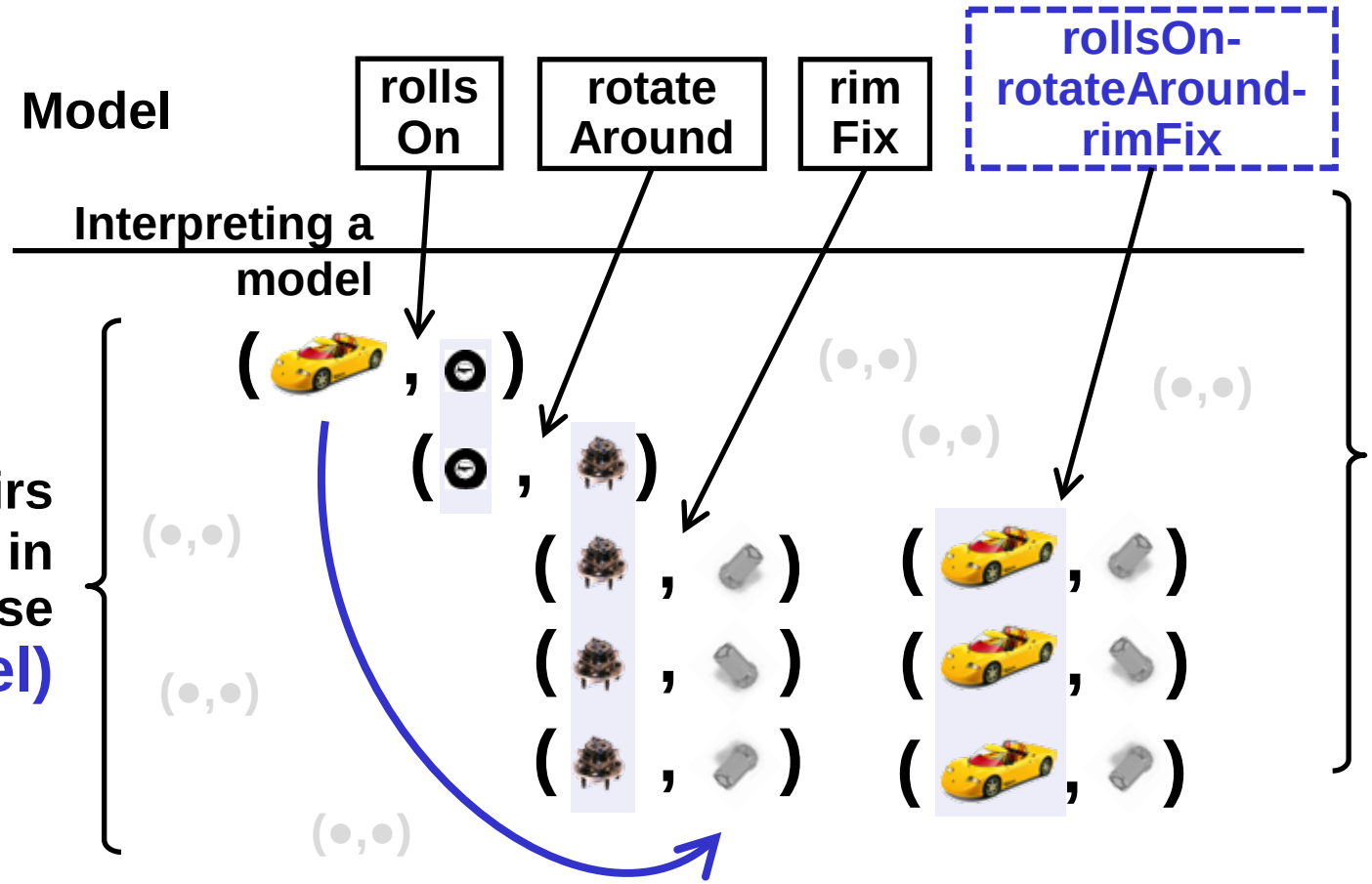


Nested connector end has property path:
(rollsOn, rotateAround, rimFix)

- **Bound values are found by "navigation" from each car.**
  - **Right end would be all lugbolts of hubs on all wheels.**
- **Don't want multiplicity on bound reference to count all LBs.**
  - **Just the ones on each wheel.**

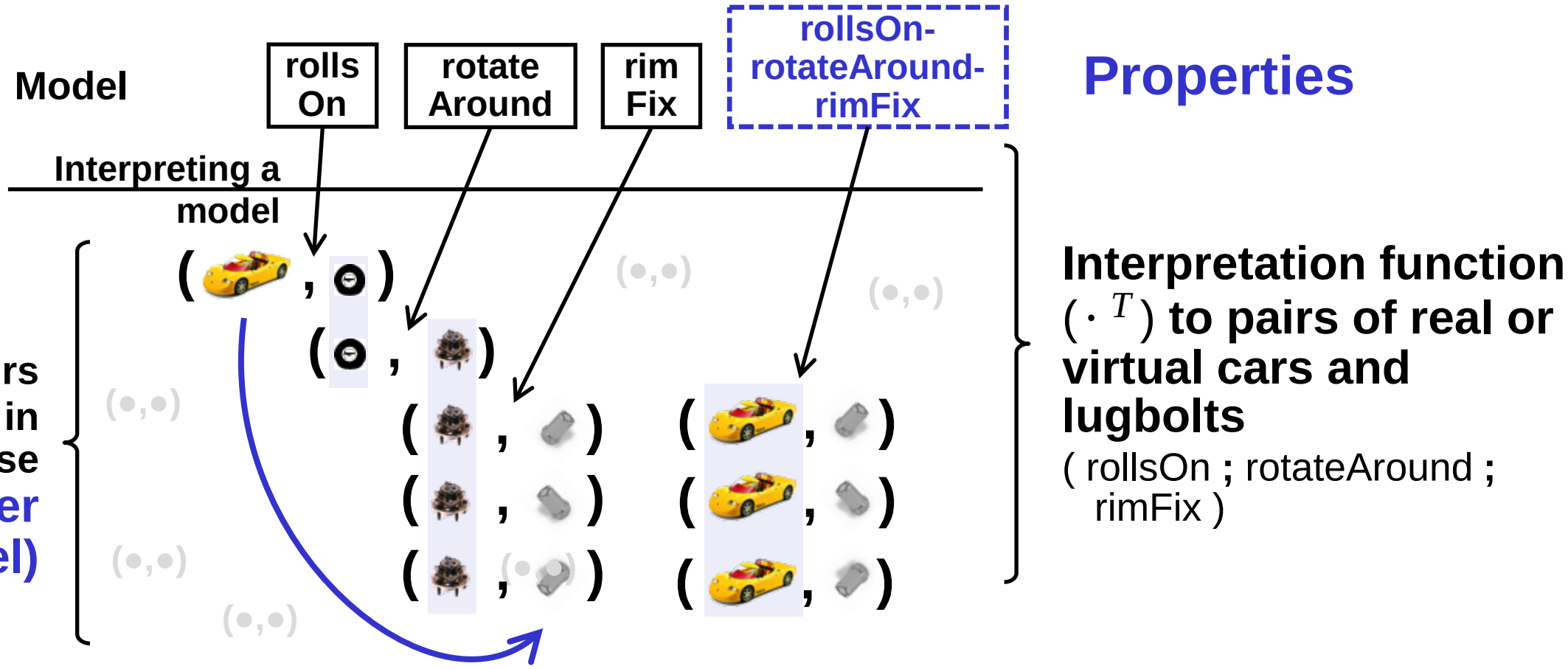
SysML 1.4 Variant WG Archive http://www.omg.org/members/sysml-rtf-wiki/doku.php?id=rtf4:groups:variant:variants_modeling

SysML 1.x PropertyPaths, Interpretation



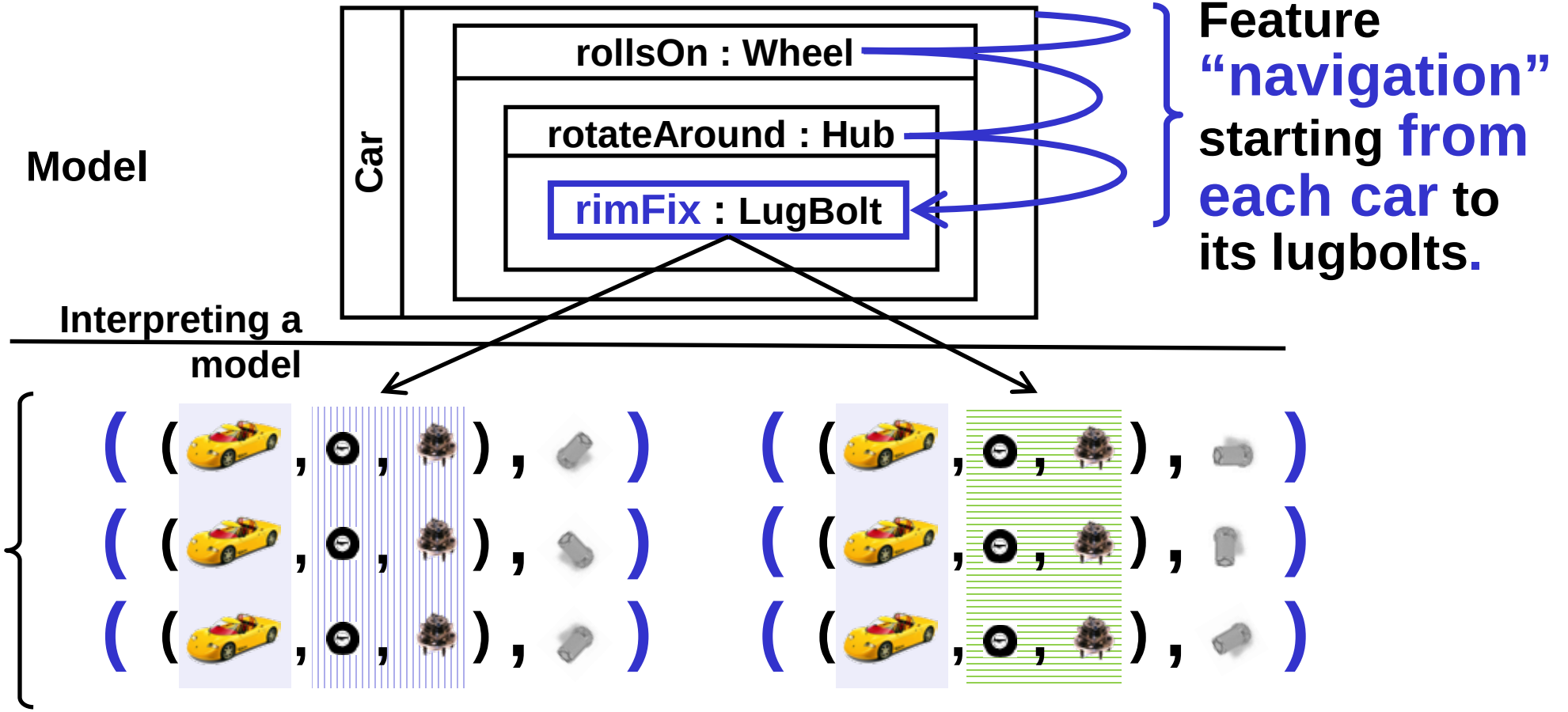
- Bound reference links cars to their lugbolts
  - It can restrict type of lugbolt.

SysML 1.x PropertyPaths, Interpretation



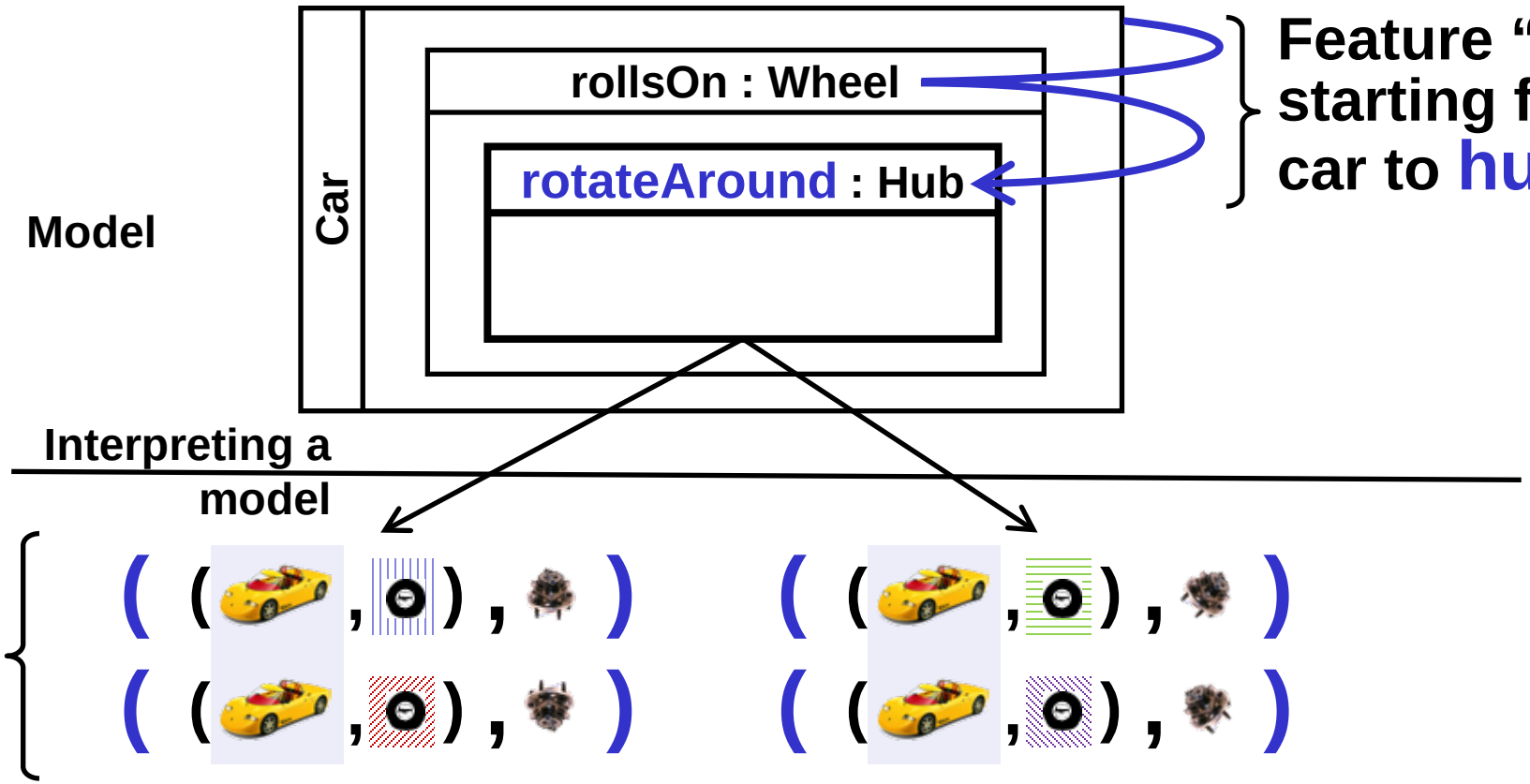
- Bound reference links cars to all their lugbolts
  - Restrictions apply to all hubs of all wheels.
  - Maybe OK for type, but probably not for multiplicity.

“Nested” Features, Interpretation



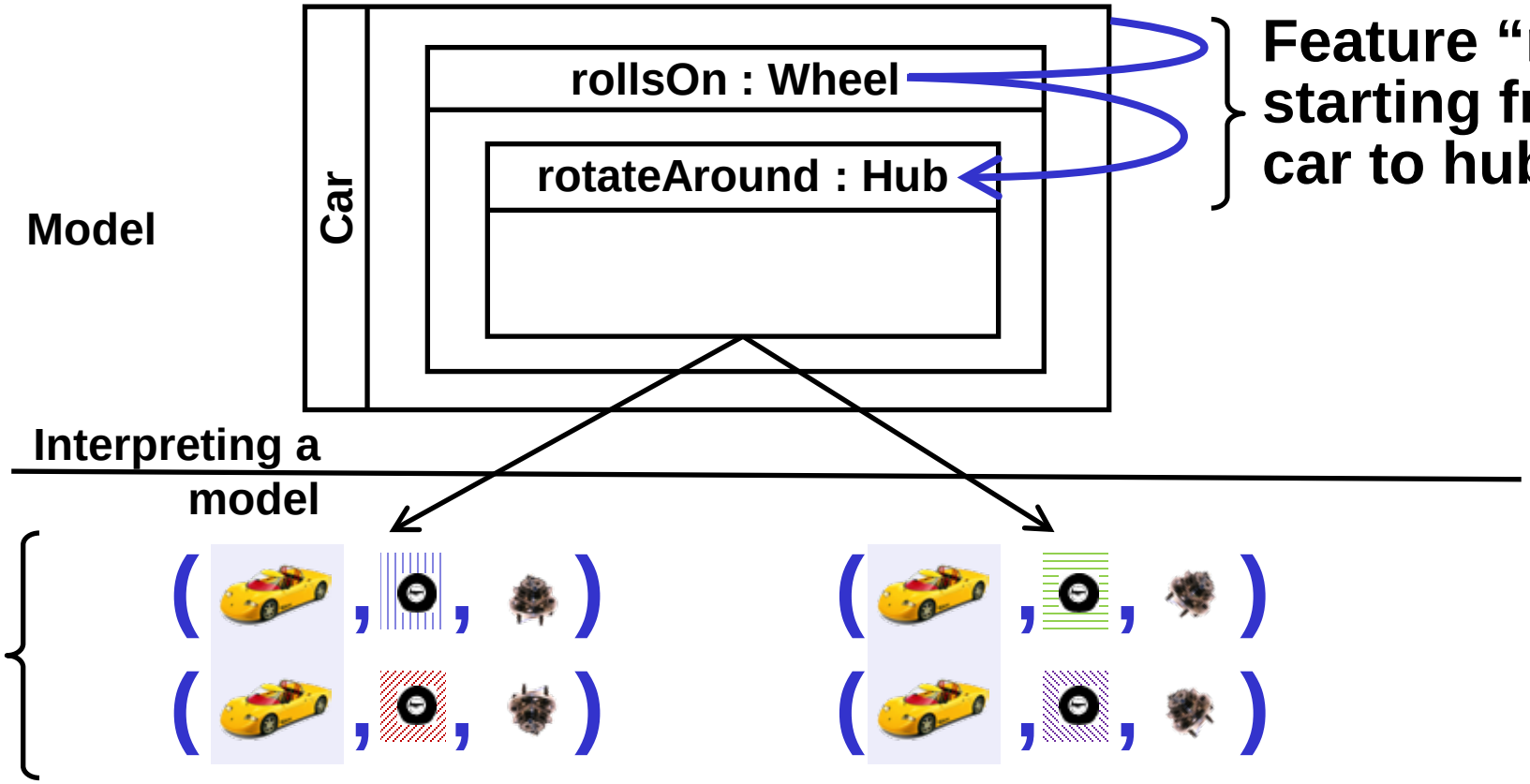
- Lugbolts paired with sequences of “navigations” to each.
  - Restrictions apply to each hub separately.
  - Works for types and multiplicity.

Less “Nested” Features, Interpretation



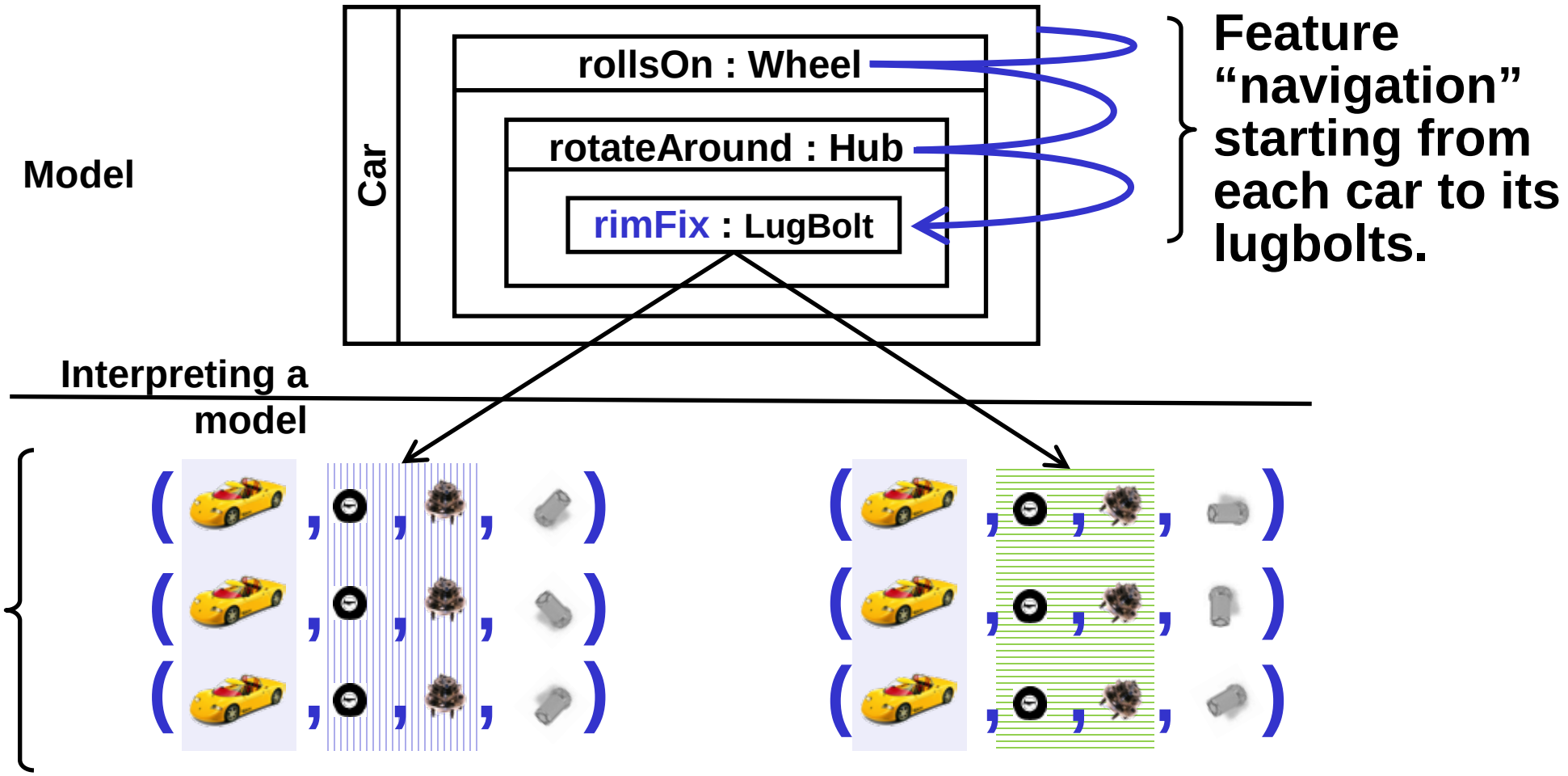
§ **Hubs paired with sequences “navigating” to each.**

SysML 2 Less “Nested” Features, Interpretation



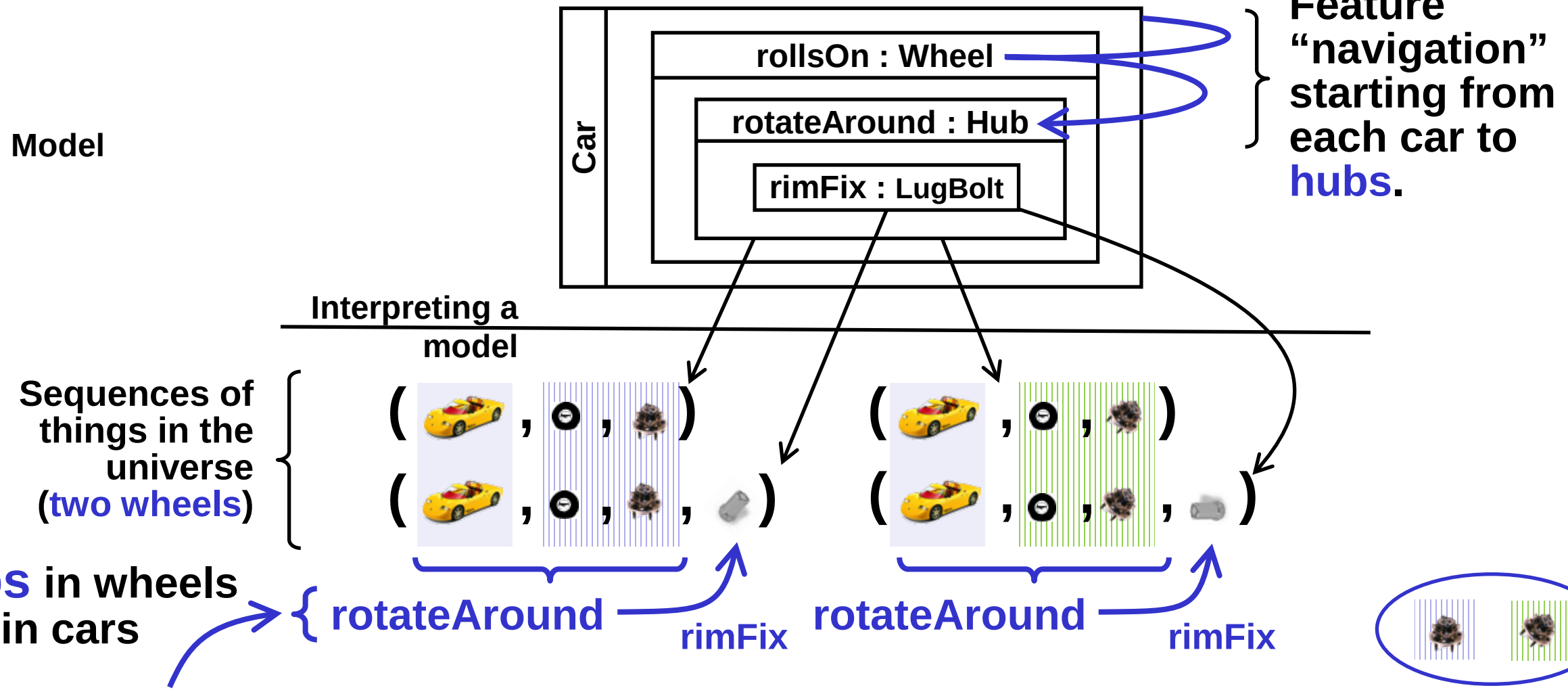
- Hubs at end of sequences “navigating” to them.
  - No nested pairs.

SysML 2 “Nested” Features, Interpretation



§ Lugbolts at end of sequences “navigating” to them.

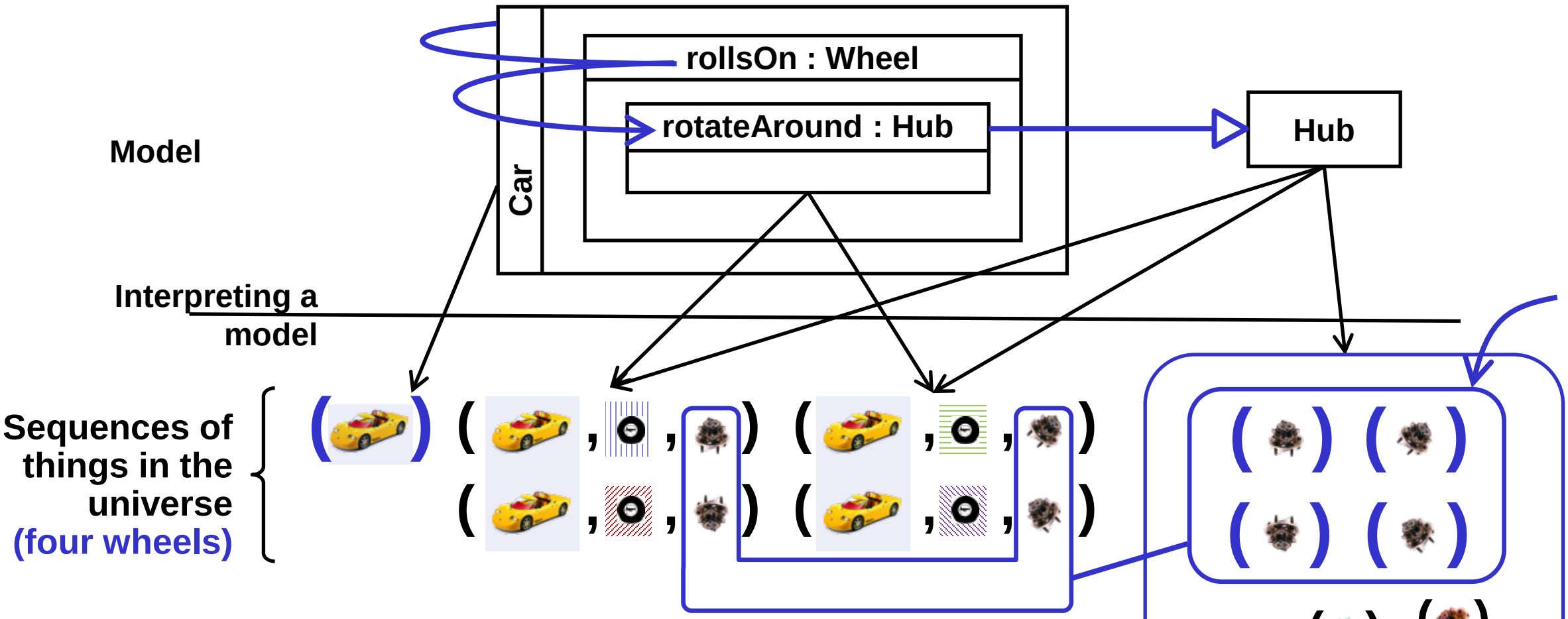
SysML 2 “Features as Classifiers” ?



§ **Nested rotateAround sequences identify a subset of hubs**

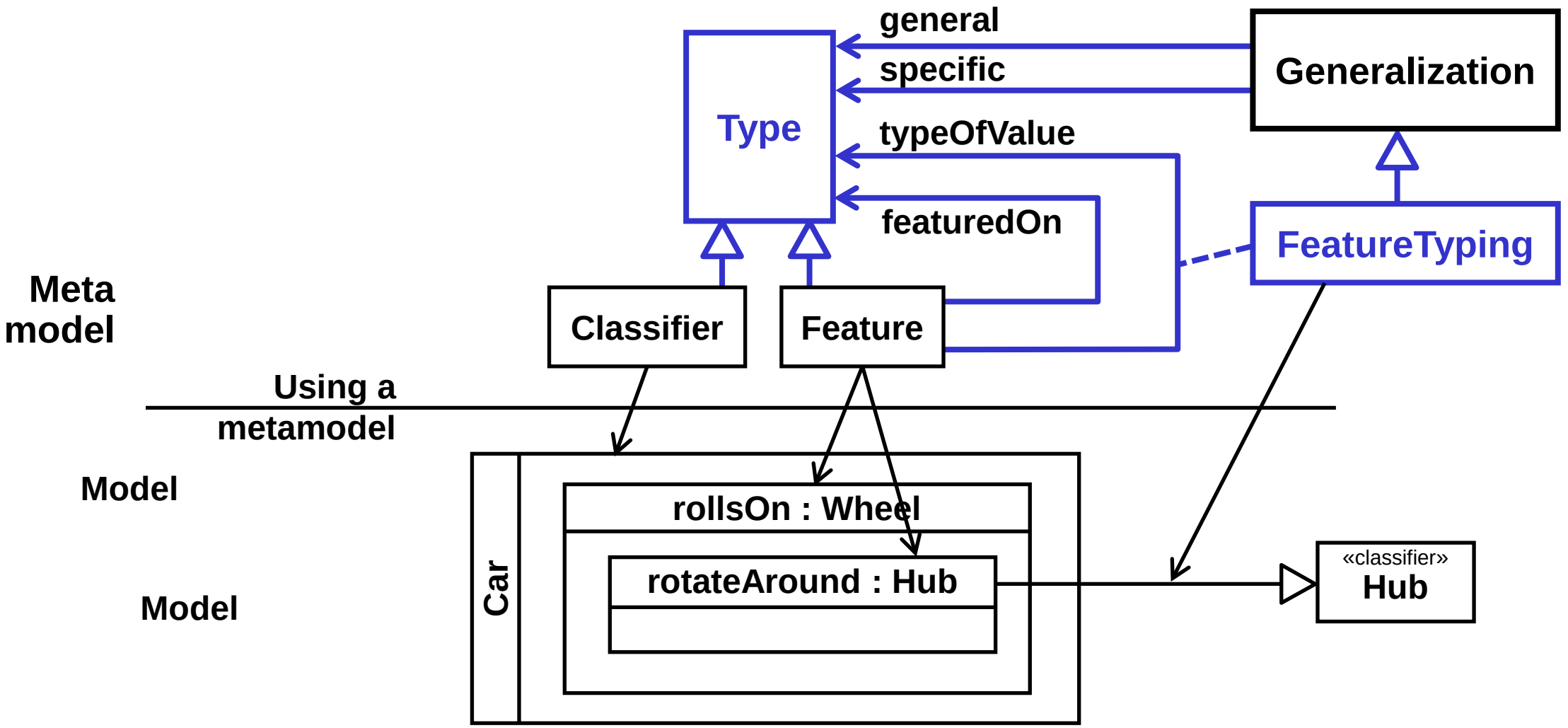
– **… without additional classes.**

SysML 2 “Features as Classifiers” ?



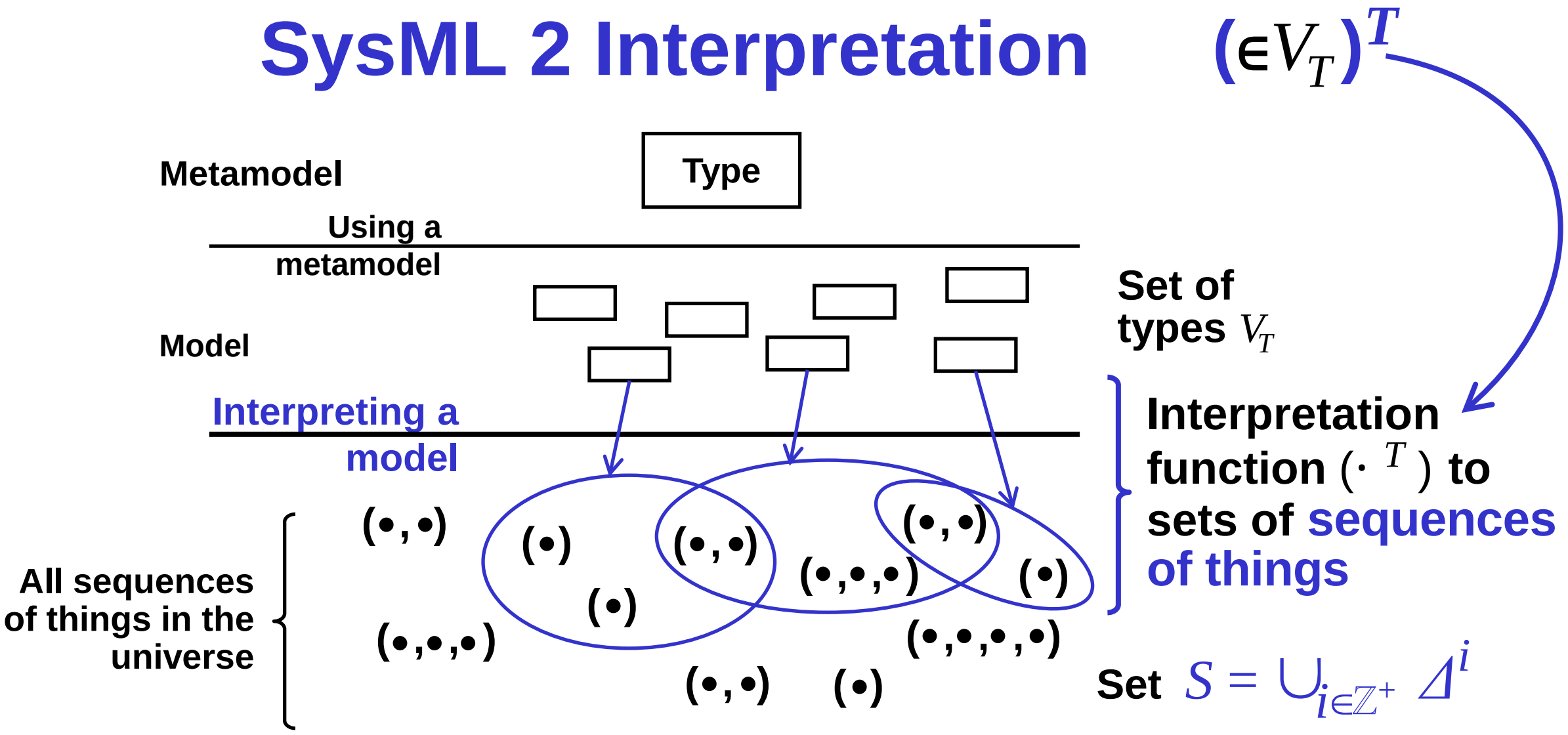
- **Classifiers interpreted as sequences**
  - **of single things, eg, hubs.**
  - **leading to those things (nested features)**

SysML 2 Features, Classifiers as Types



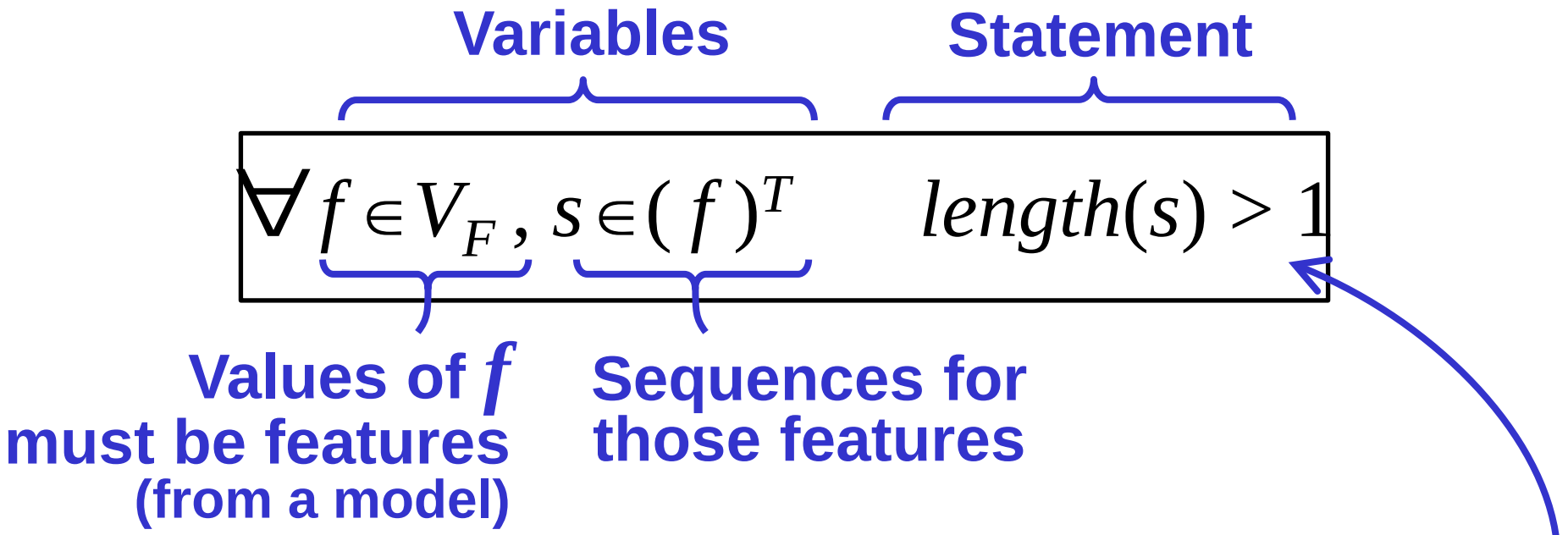
- **Metamodel : Feature, Classifier are disjoint**
- **Model : Features, Classifiers are not.**

SysML 2 Interpretation



- Links model elements to sets of sequences of things in the universe.

Sequence Interpretation, Features

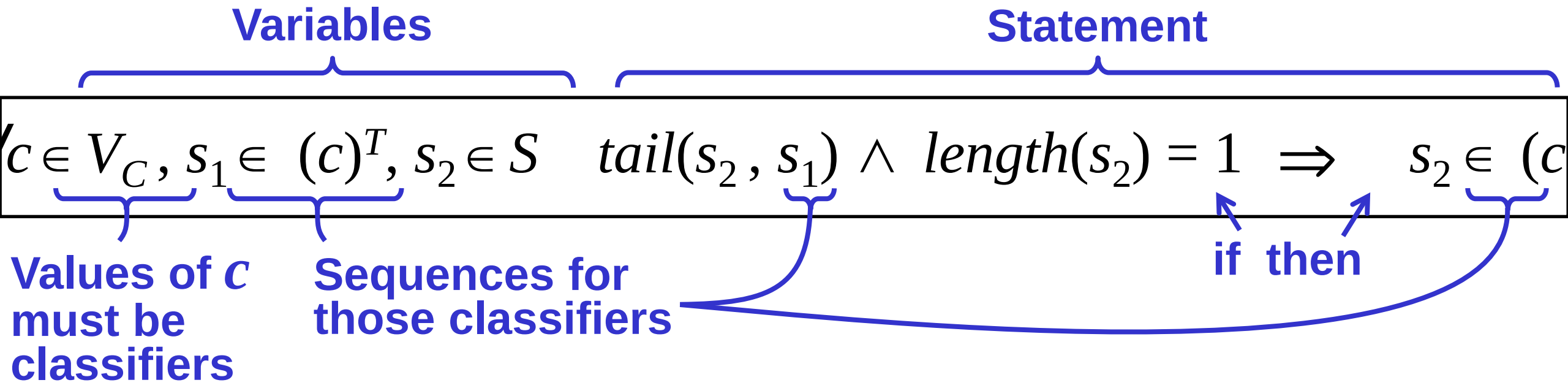


7.3.4 Features 7.3.4.4 Semantics

1. The interpretations of features must have length greater than one.

- **Feature sequences are longer than one.**
  - **They relate (“lead”/”navigate” from/to) things in the universe.**

Sequence Interpretation, Classifiers



7.3.3 Classifiers 7.3.3.4 Semantics

1. If the interpretation of a Classifier includes a sequence, it also includes the 1-tail of that sequence.

- **Classifier sequences longer than one (= feature sequences) imply the ending 1-sequence is included.**

Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- Solution
  - The “S” Words
  - Standardizing Semantics
  - Conformance = Classification
  - Formalizing Semantics (ie, a little math)
  - SysML 2 Semantics
- **The “O” Word**
- Summary

The “O” Word

- **Has many meanings**
  - **Can spend more time defining it than doing it.**
- **Two meanings used in this presentation:**
  1. **Start with the things being modeled (real, desired, imagined, simulated, etc).**
  2. **Group (classify) those things by their commonalities.**

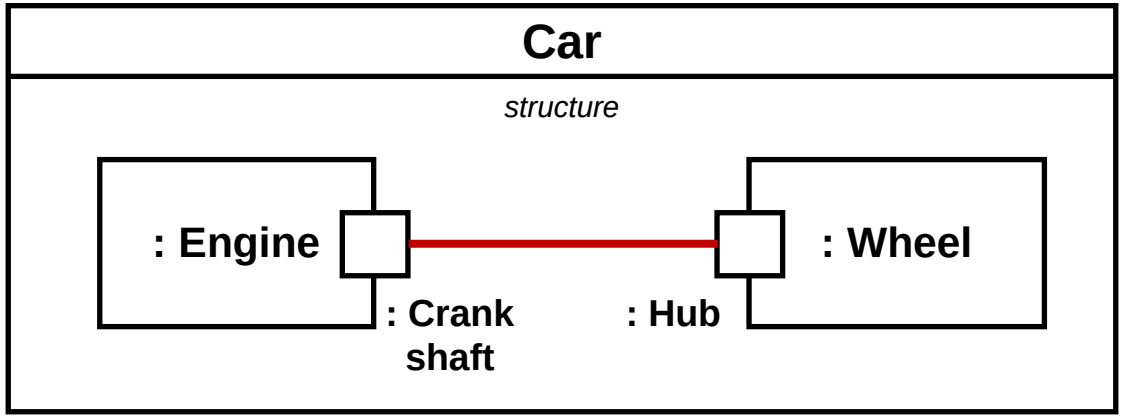
OWL (Web Ontology Language)

- **Interchange standard for a kind of description logic (DL).**
  - **Arrived at after decades of (early, not statistical) AI research**
    - **Formalizing commonly needed information/knowledge**
  - **Started without logic (eg, "semantic nets")**
  - **Eventually reduced to named FOL patterns**
    - **OWL = SROIQ$^{(D)}$**
- **DL letters**
  - **I = Inverse properties** $\forall x, y \quad p_1(x, y) \Leftrightarrow p_2(y, x)$
  - **S includes**
    - **Concept intersection** $\forall x \; c_1(x) \Leftrightarrow c_2(x) \wedge c_3(x)$
    - **Transitive roles** $\forall x, y, z \;\; p(x, y) \wedge p(y, z) \;\; \Rightarrow \;\; p(x, z)$

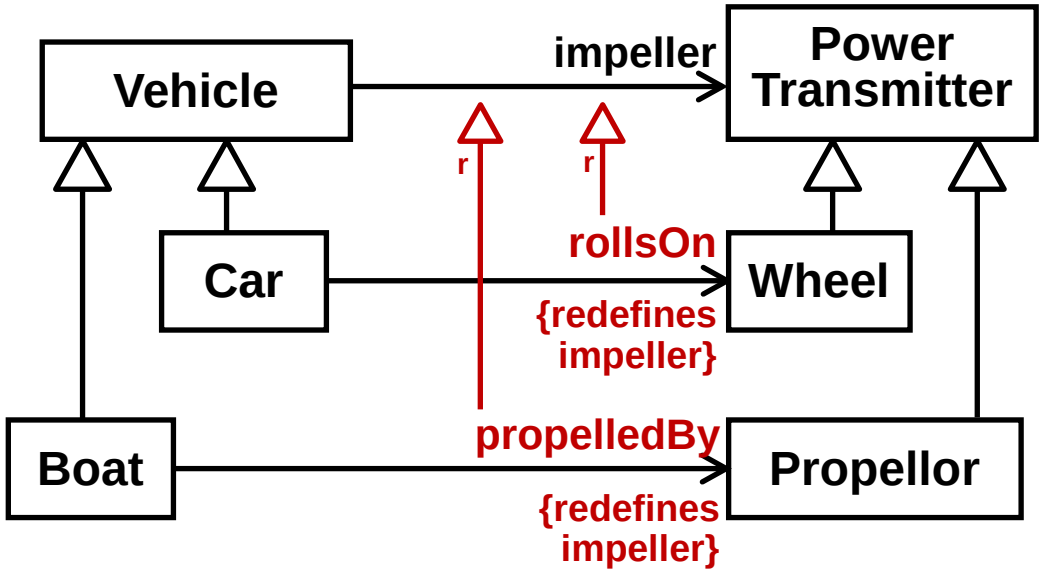
UML/SysML1 Outside SROIQ/OWL

- **Connectors between ports and other nested properties.**

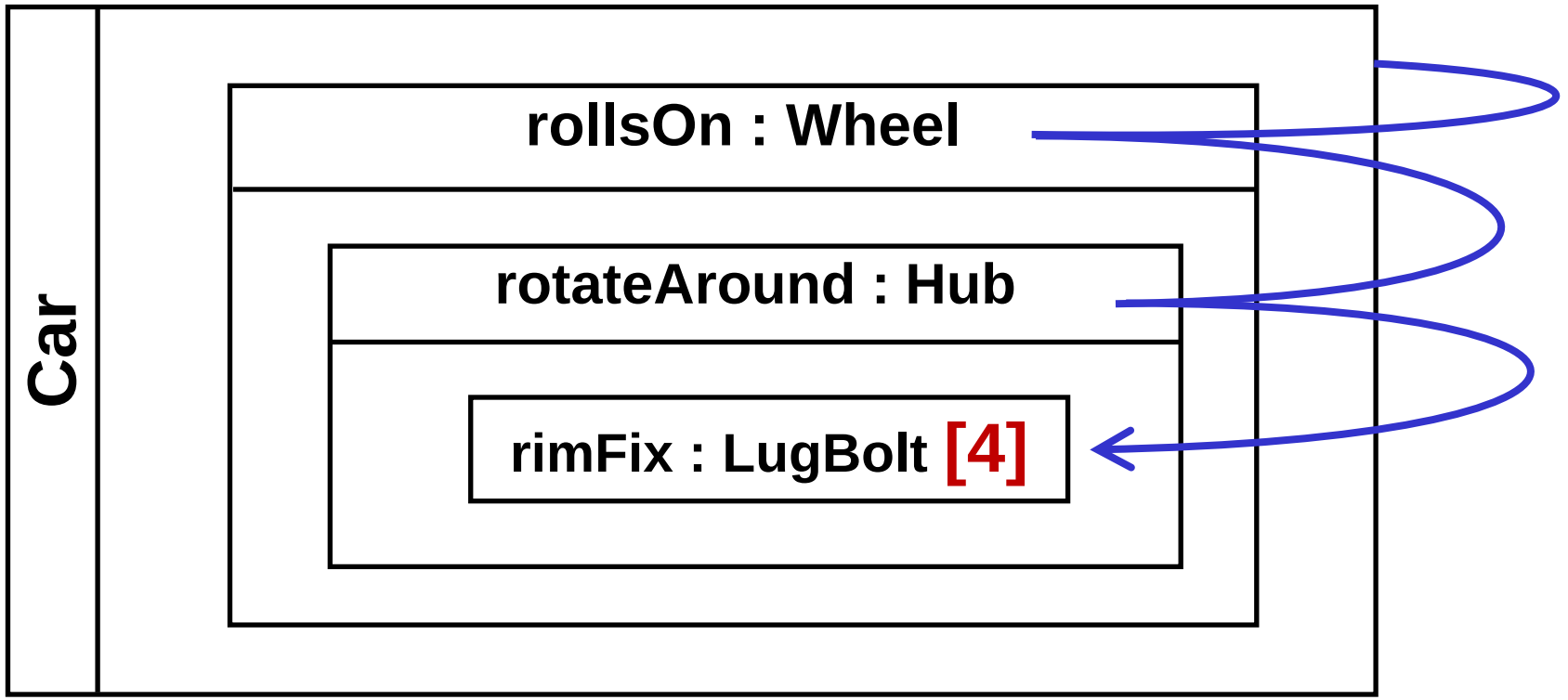
https://www.nist.gov/publications/reasoning-manufacturing-part-part-examples-owl-2



- **Property redefinition that “changes” the name.**



SST Outside SROIQ/OWL



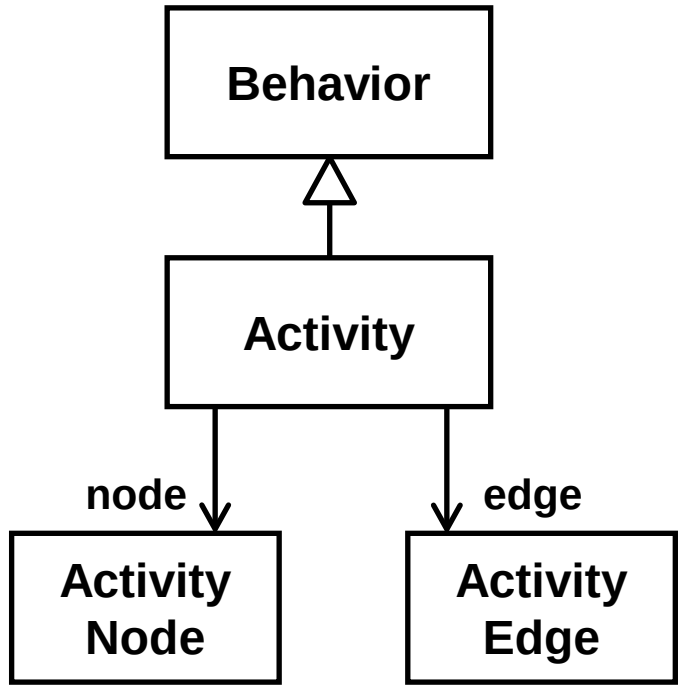
- Redefinition of multiplicity on nested features.
  - Can't restrict number of lugbolts on each hub.
  - Can redefine multiplicity (and type) for all lugbolts.

OWL ≠ “S” (or “O”)

```
Declaration( Class( Behavior ) )
Declaration( Class( Activity ) )
Declaration( Class( ActivityNode ) )
Declaration( Class( ActivityEdge ) )
Declaration( ObjectProperty( node ) )
Declaration( ObjectProperty( edge ) )

SubClassOf( Activity Behavior )
ObjectPropertyDomain ( node Activity )
ObjectPropertyRange  ( node ActivityNode )
ObjectPropertyDomain ( edge Activity )
ObjectPropertyRange  ( edge ActivityEdge )
```

OWL



UML Metamodel (M2)

- Is this UML semantics?
  - No, it’s syntax specified in an “S/O” language.

Overview

- Motivation / Problem : Analysis
  - Systems Engineering
  - Modeling Languages
- Solution
  - The “S” Words
  - Standardizing Semantics
  - Conformance = Classification
  - Formalizing Semantics (ie, a little math)
  - SysML 2 Semantics
- The “O” Word
- **Summary**

Summary, SE and Analysis

- System engineers interact with domain engineers
  - who regularly use mathematical tools to predict system behavior.
  - SEs need these tools also to check domain analysis results.
- Language designers and analysis tool builders have
  - expectations for system construction / operation ...
  - … coordinated through a standards specifications.

Summary, Syntax & Semantics

- Syntax specifies models
- Semantics + models specify real or virtual things
  - Enables checking those things against the model.
  - Conformance (checking) = classification (yes/no).
- Specifying semantics
  - Constraints that (kinds of) model elements place on classifying (pairs of) things in a hypothetical universe.

Summary, SysML 2

- **Semantic framework, motivation**
  - **Classifying sequences of things in a hypothetical universe ...**
  - **… to model subsets of things reached by feature "navigation" …**
  - **… without additional classes. Facilitates variation modeling.**
- **Features and Classifiers**
  - **Features interpreted as sequences longer than one.**
  - **Classifiers interpreted as sequences of exactly one thing + …**
  - **… all feature sequences ending in those things.**
  - **Enables features to be "classifers" for other ("nested") features.**
  - **Kinds of feature values (typing) = Generalization**

Other Information

- **OWL 2 Direct Semantics**
  - **https://www.w3.org/TR/owl2-direct-semantics/**
- **Introduction to Reasoning**
  - **Section 3.1 in Bock, et al, "Evaulating Reasoning Systems," NISTIR 7310 https://www.nist.gov/publications/evaluating-reasoning-systems**
- **SysML 1.4 Variant WG Archive**
  - **http://www.omg.org/members/sysml-rtf-wiki/doku.php?id=rtf4:groups:variant:variants_modeling**
  - **Scroll down for literature and presentations.**
  - **Discussion deck: http://tinyurl.com/ybxlc2wy**
    - **Bound references on slides 12-44.**