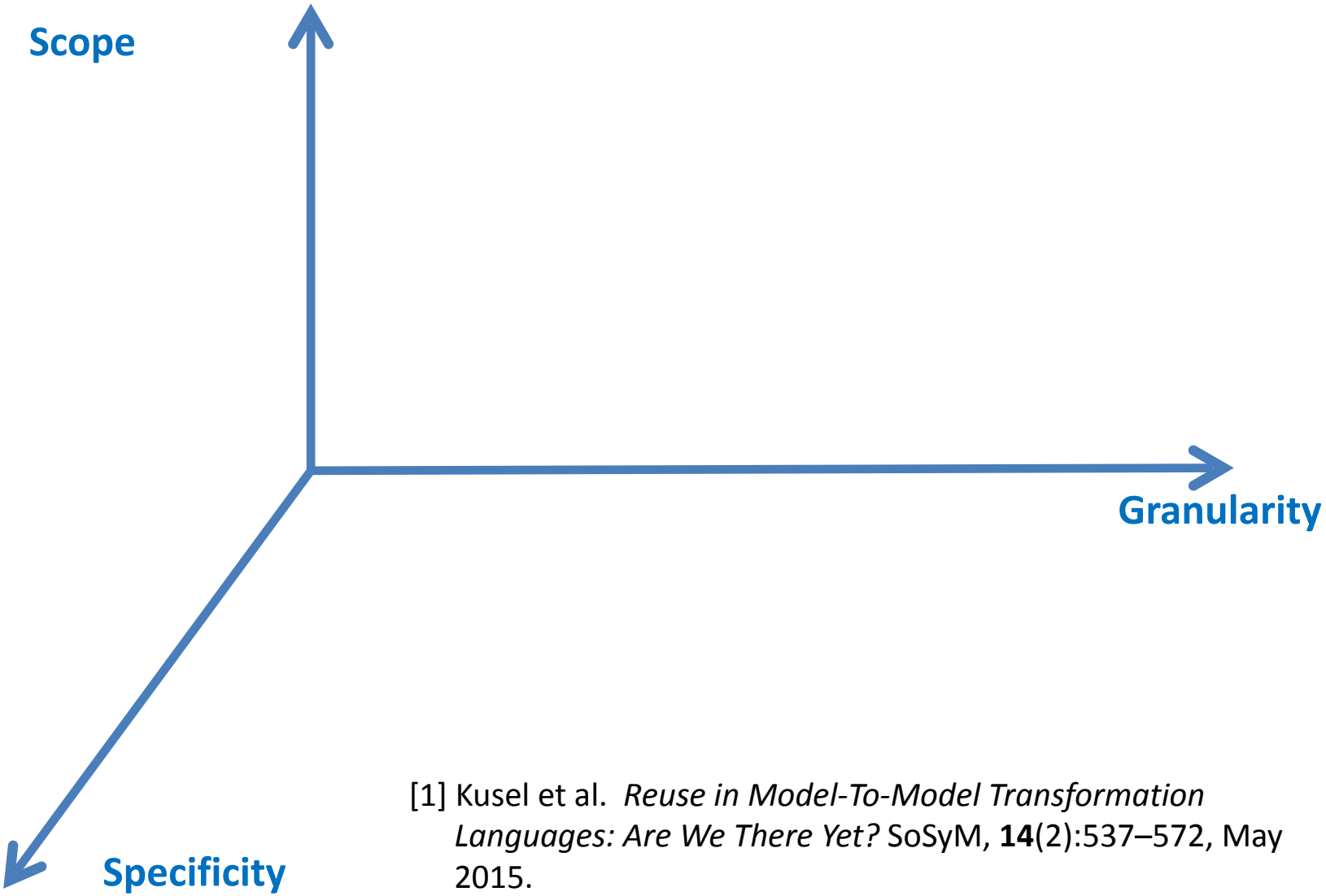
Featured Model Types

Towards Systematic Reuse in Modelling Language Engineering

Gilles Perrouin, Moussa Amrani, Mathieu Acher, Benoît Combemale, Axel Legay, Pierre-Yves Schobbens

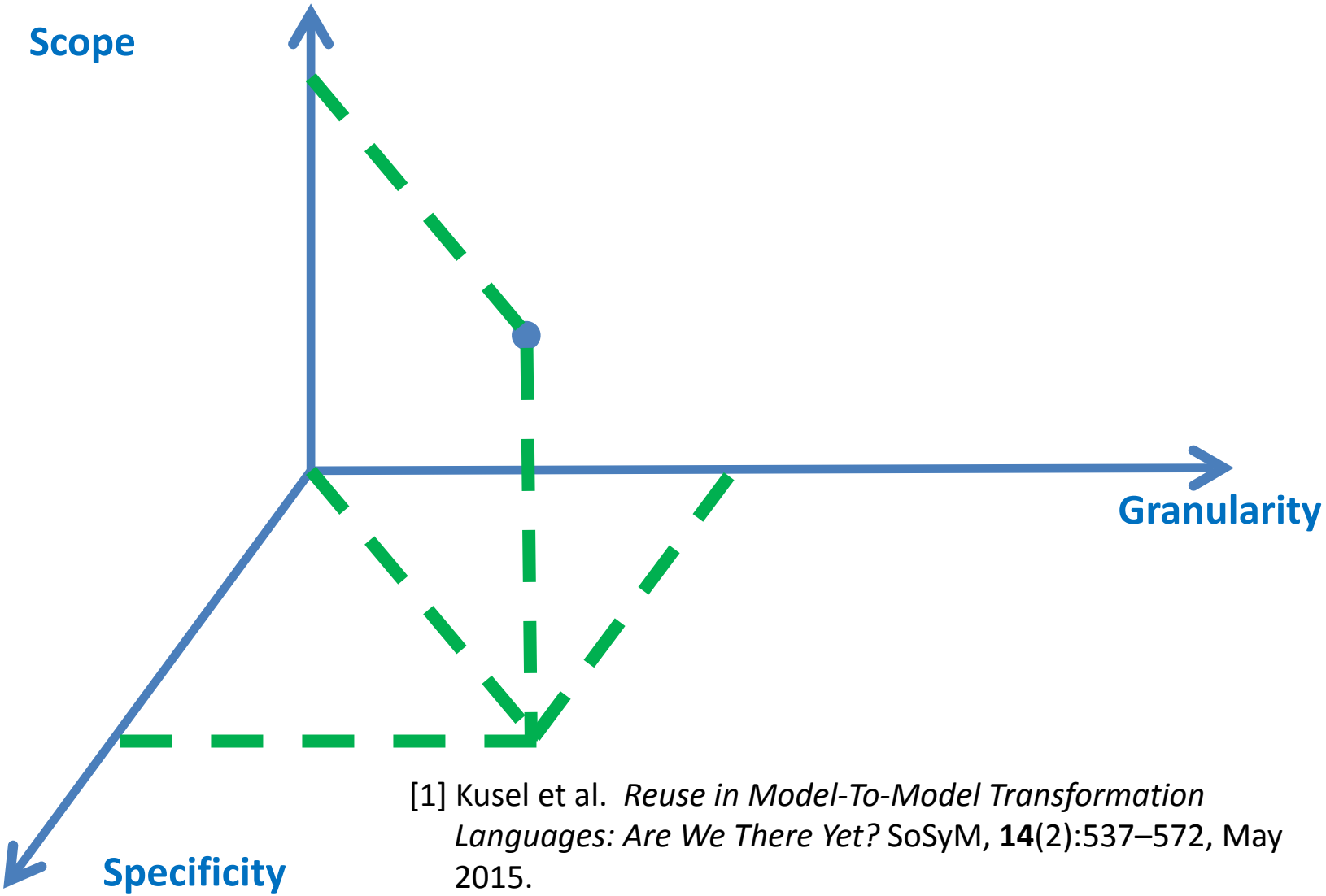
MISE@ICSE, Austin
May 16, 2016

Reuse Dimensions [1]



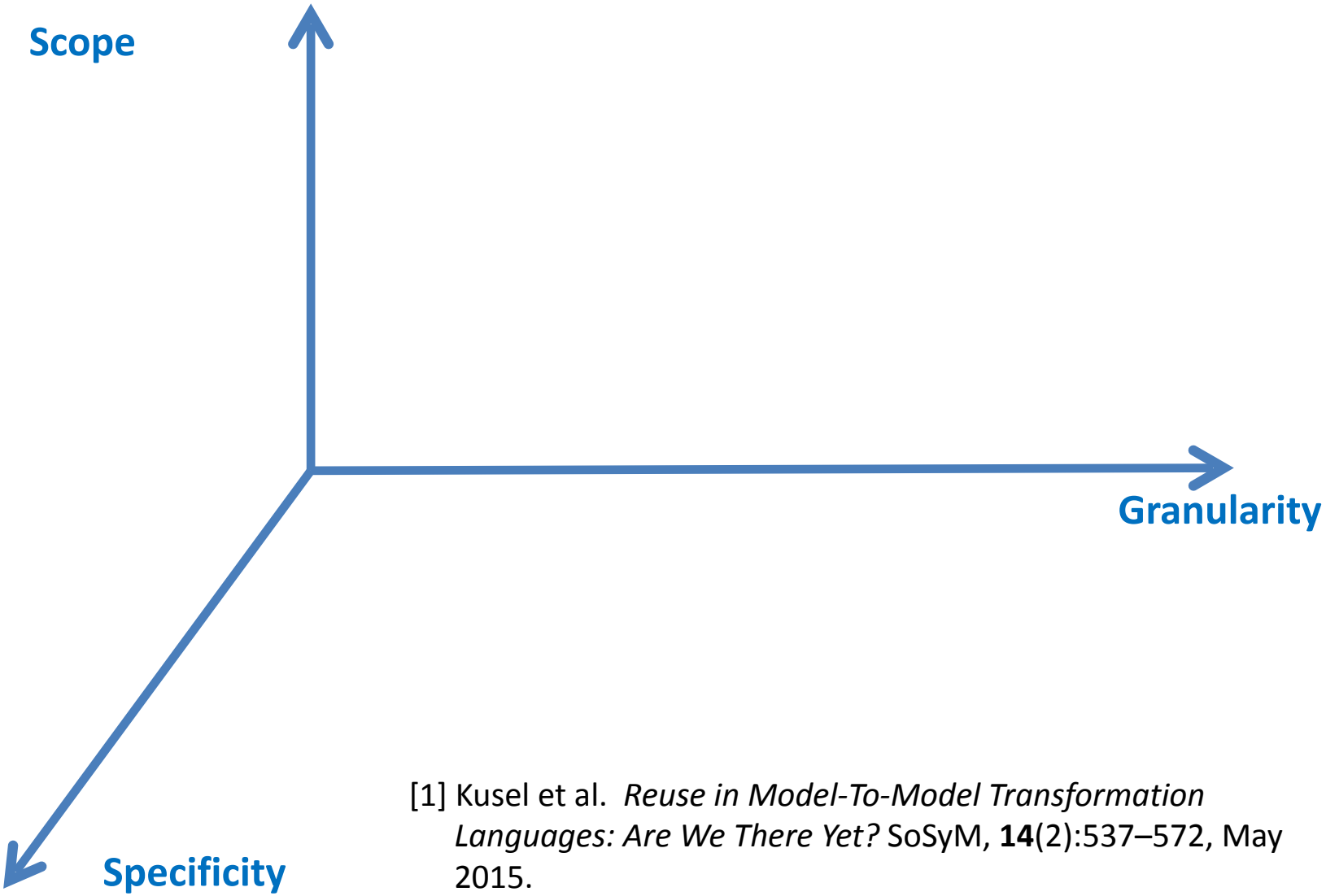
[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2):537–572, May 2015.

Reuse Dimensions [1]



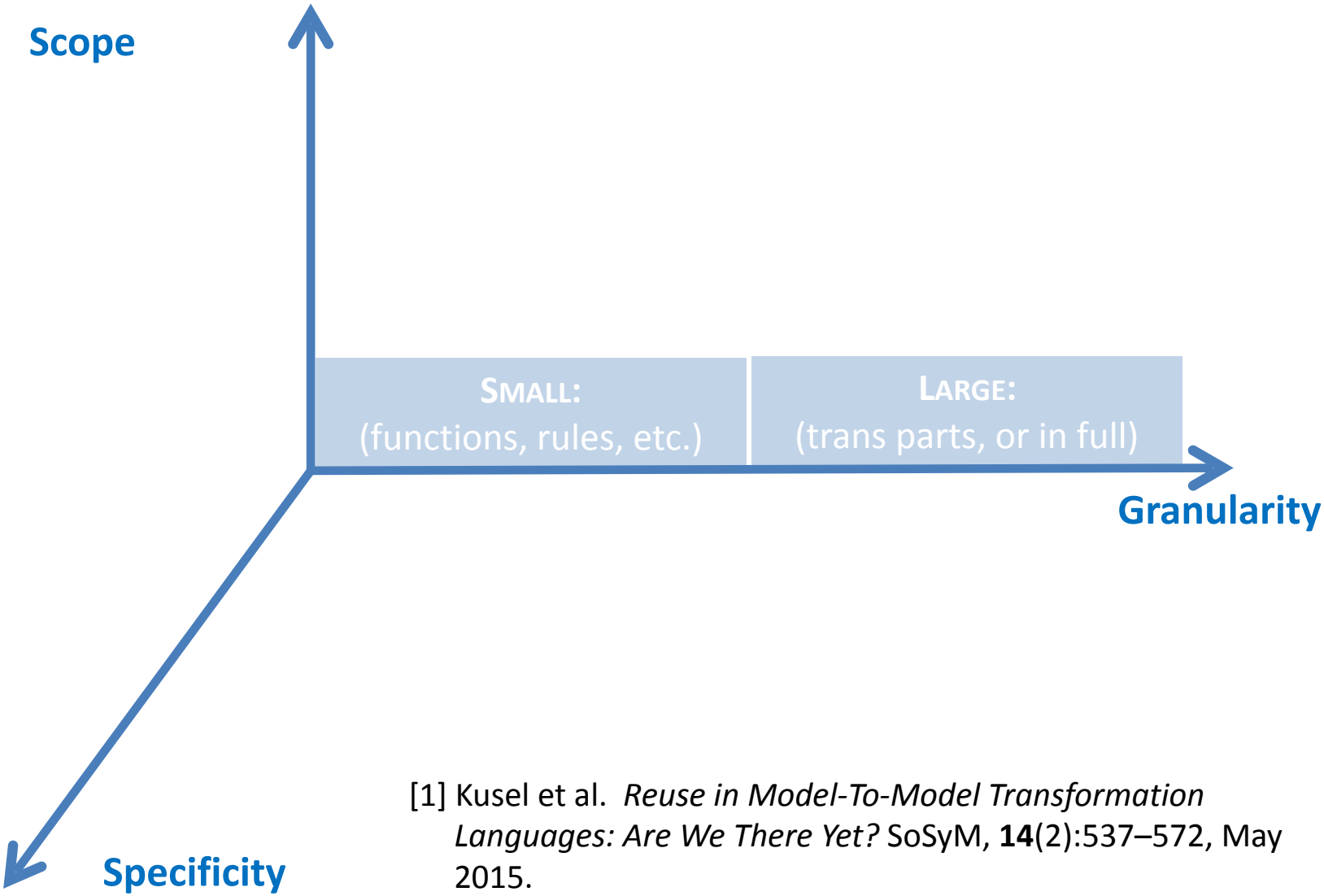
[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2):537–572, May 2015.

Reuse Dimensions [1]



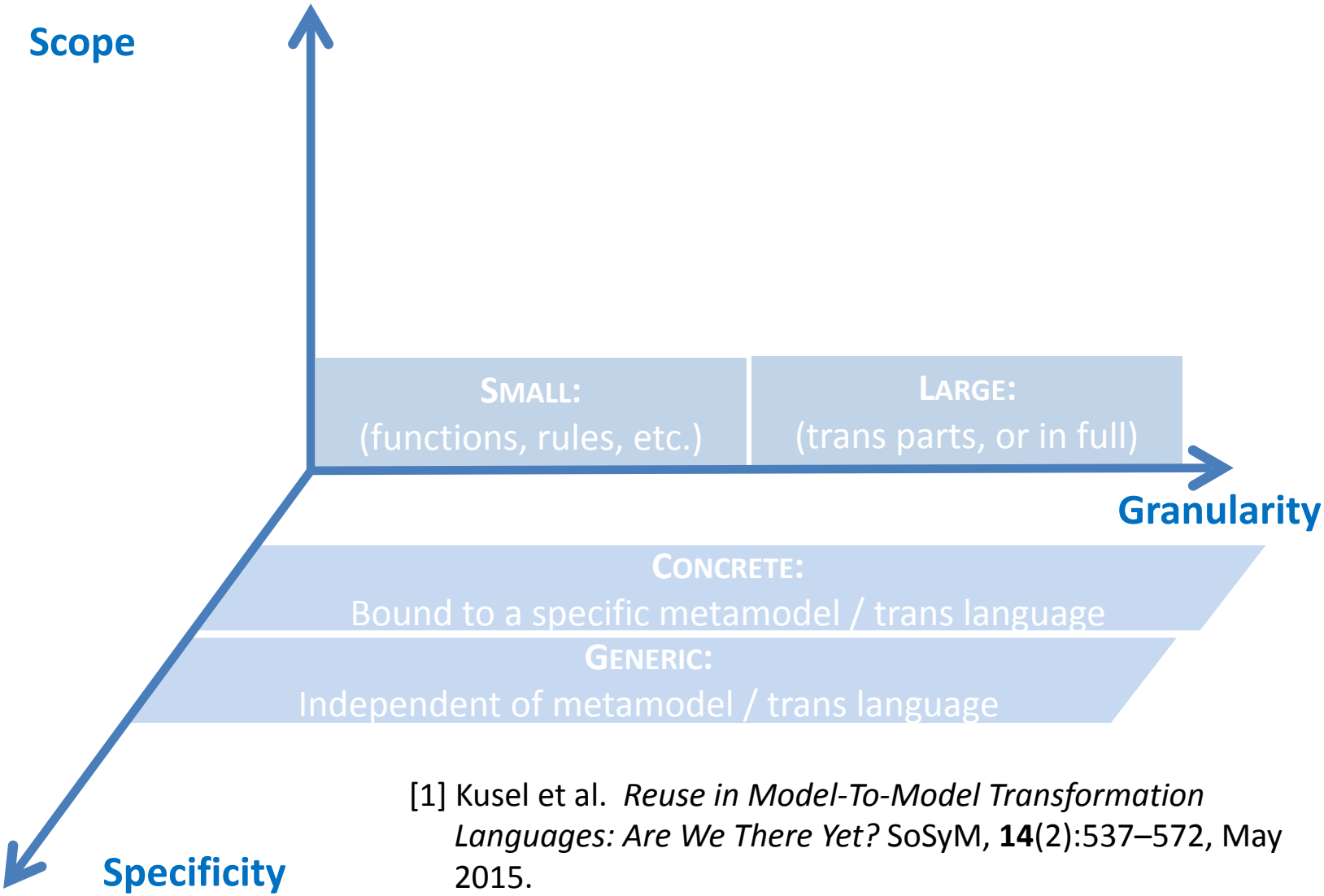
[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2):537–572, May 2015.

Reuse Dimensions [1]



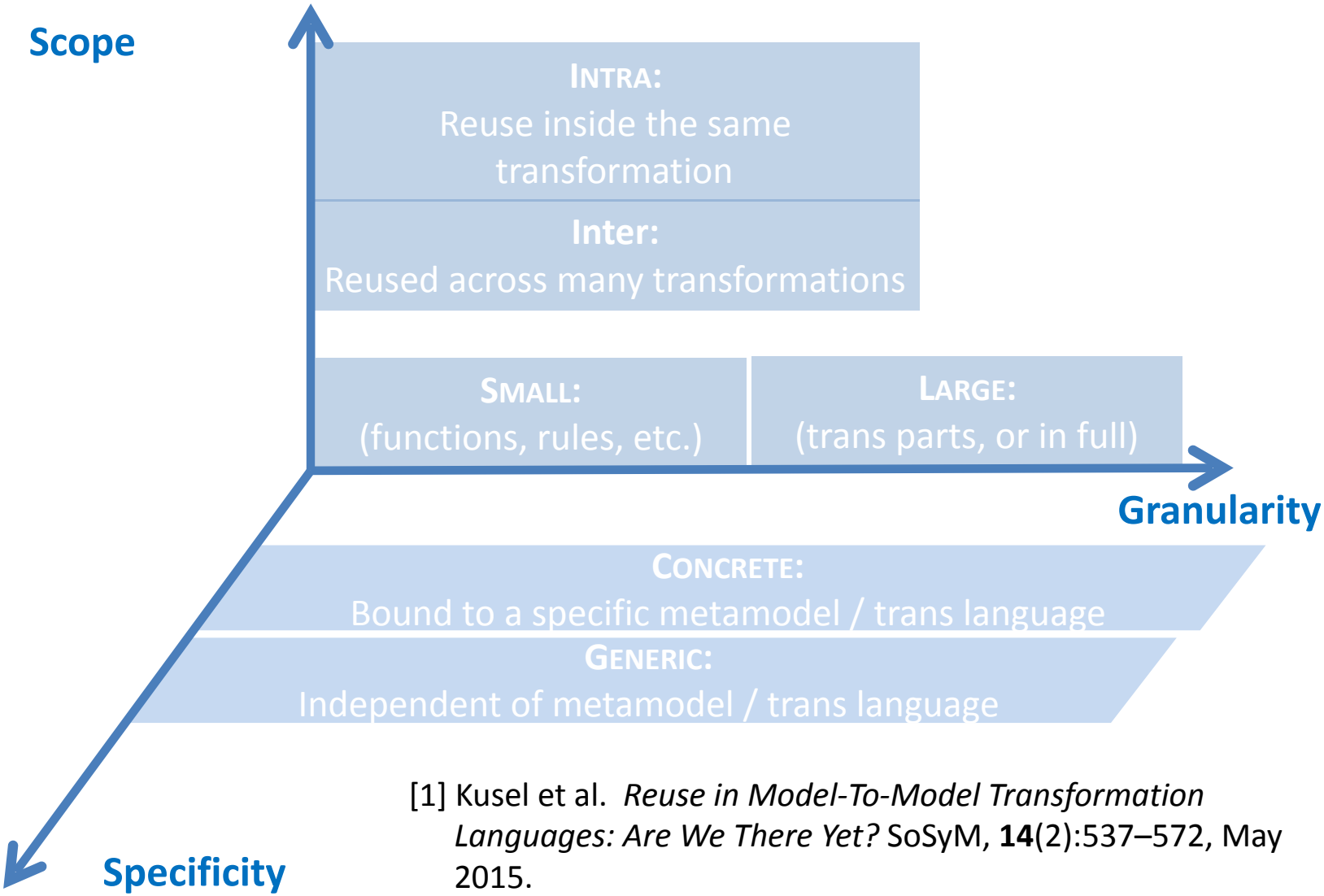
[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2):537–572, May 2015.

Reuse Dimensions [1]



[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2):537–572, May 2015.

Reuse Dimensions [1]



[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2):537–572, May 2015.

Model Reuse Strategies

- By adapting the transformation to be reused

- By adapting the source metamodel

[1] Salay, Rick and Famelis, Michalis and Rubin, Julia and Di Sandro, Alessio and Chechik, Marsha. Lifting Model Transformations To Product Lines. ICSE, 2014.
[2] de Lara, Juan and Guerra, Esther and Cuadrado, Jesus Sanchez.A-posteriori typing for Model-Driven Engineering. MoDELS 2015.
[3] Tisi, Massimo and Jouault, Frédéric and Fraternali, Piero and Ceri, Stefano and Bézivin, Jean. On The Use of Higher-Order Transformations. MDA-FA, 2009.
[4] Guy, Clément and Combemale, Benoît and Derrien, Steven and Steel, James and Jézéquel, Jean-Marc. On Model Subtyping. ECMFA, 2012.
[5] Moha, Naouel and Mahé, Vincent and Barais, Olivier and Jézéquel, Jean-Marc. Generic Model Refactorings. MoDELS 2009.
[6] Sen, Sagar and Moha, Naouel and Mahé, Vincent and Barais, Olivier and Baudry, Benoît and Jézéquel, Jean-Marc. Reusable model transformations. SoSyM, **11**(1), 2010.

Issues [1]

[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2): 537–572, May 2015.

Issues [1]

I1 – Insufficient abstraction from metamodel(s) to support metamodel-independent reuse

[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2): 537–572, May 2015.

Issues [1]

I1 – Insufficient abstraction from metamodel(s) to support metamodel-independent reuse

By ***generalization***: decouple transformation logic from type info
By ***simplification***: expose interface; hide realisation

[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2): 537–572, May 2015.

Issues [1]

I1 – Insufficient abstraction from metamodel(s) to support metamodel-independent reuse

By ***generalization***: decouple transformation logic from type info
By ***simplification***: expose interface; hide realisation

I2 – Lack of repositories for simplifying artifacts selection, at coarse- (e.g., full transfos) and fine-grained (e.g., functions) levels

[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2): 537–572, May 2015.

Issues [1]

I1 – Insufficient abstraction from metamodel(s) to support metamodel-independent reuse

By ***generalization***: decouple transformation logic from type info
By ***simplification***: expose interface; hide realisation

I2 – Lack of repositories for simplifying artifacts selection, at coarse- (e.g., full transfos) and fine-grained (e.g., functions) levels

I3 – Lack of meta-information for selecting appropriate reusable elements without knowing the transformation's internal

[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2): 537–572, May 2015.

Issues [1]

I1 – Insufficient abstraction from metamodel(s) to support metamodel-independent reuse

By ***generalization***: decouple transformation logic from type info
By ***simplification***: expose interface; hide realisation

I2 – Lack of repositories for simplifying artifacts selection, at coarse- (e.g., full transfos) and fine-grained (e.g., functions) levels

I3 – Lack of meta-information for selecting appropriate reusable elements without knowing the transformation's internal

Provide documentation, pre-conditions,

test models, formal requirements, etc

[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2): 537–572, May 2015.

Issues [1]

I1 – Insufficient abstraction from metamod
metamodel-independent reuse

- By ***generalization***: decouple t
- By ***simplification***: expose

I2 – Lack of reposi tion, at coarse-
(e.g., full nctions) levels

I3 – Lack of m g appropriate reusable
elements w transformation's internal

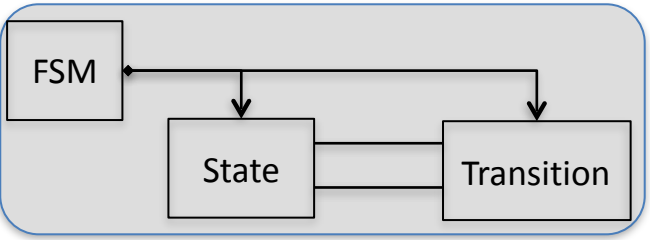
- entation, pre-conditions,
- te els, formal requirements, etc

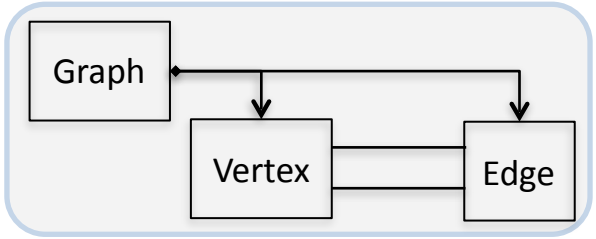
Systematic reuse is not there yet

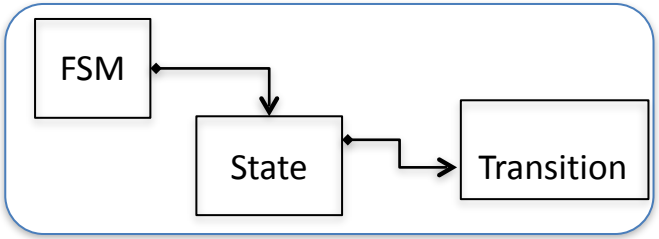
[1] Kusel et al. *Reuse in Model-To-Model Transformation Languages: Are We There Yet?* SoSyM, **14**(2): 537–572, May 2015.

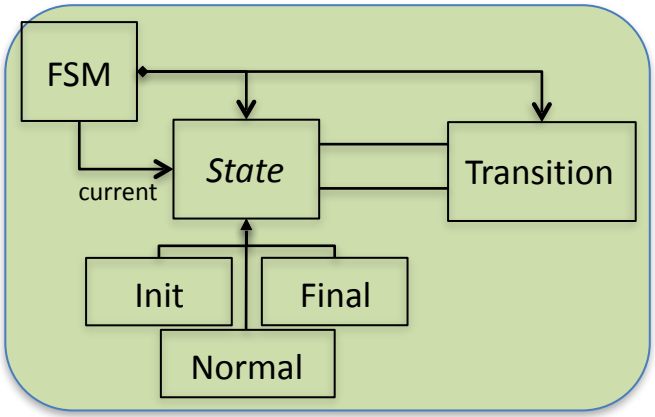
Systematising model reuse by adopting a product line approach

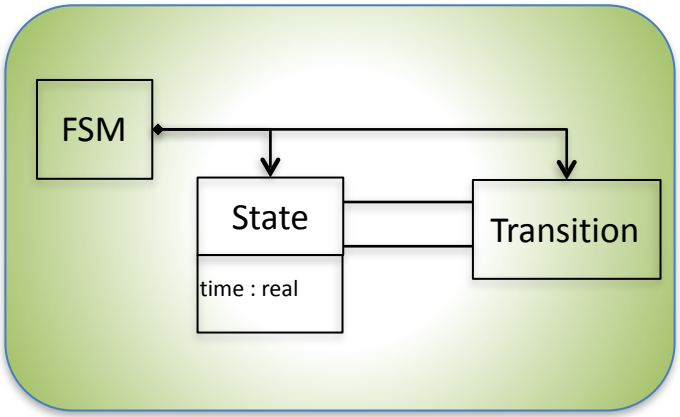
Variations over an FSM



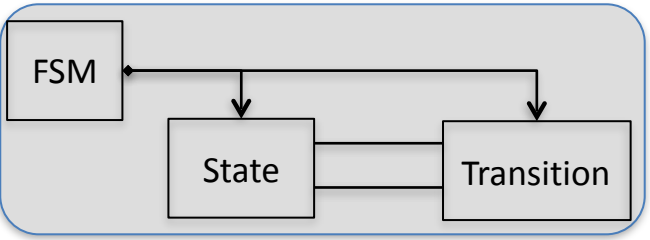


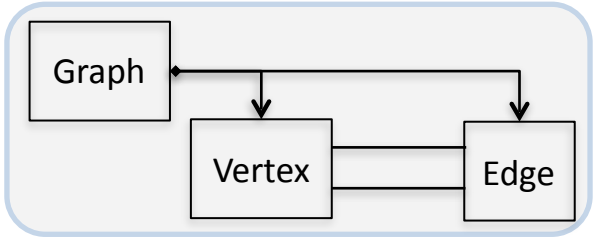


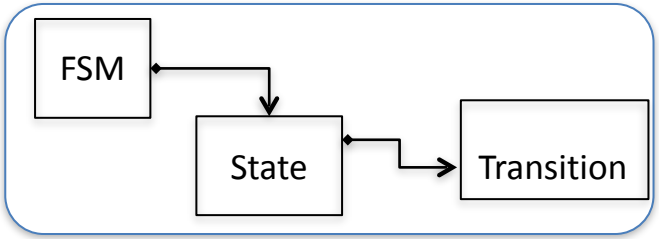




Variations over an FSM

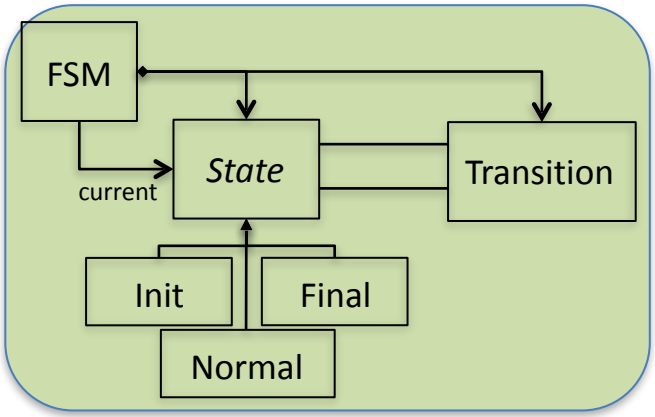






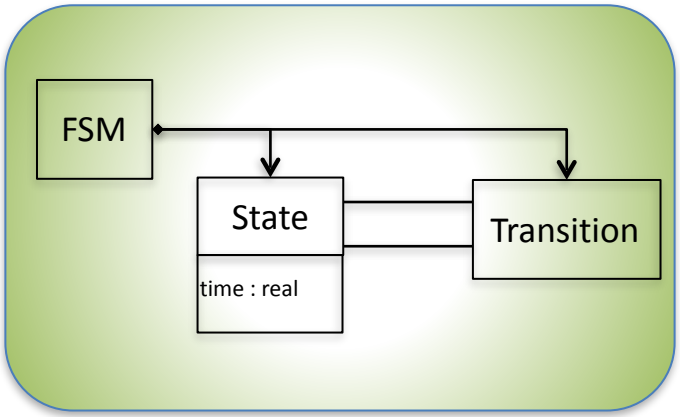
Applicable transformations:

`minimize() : FSM`



Applicable transformations:

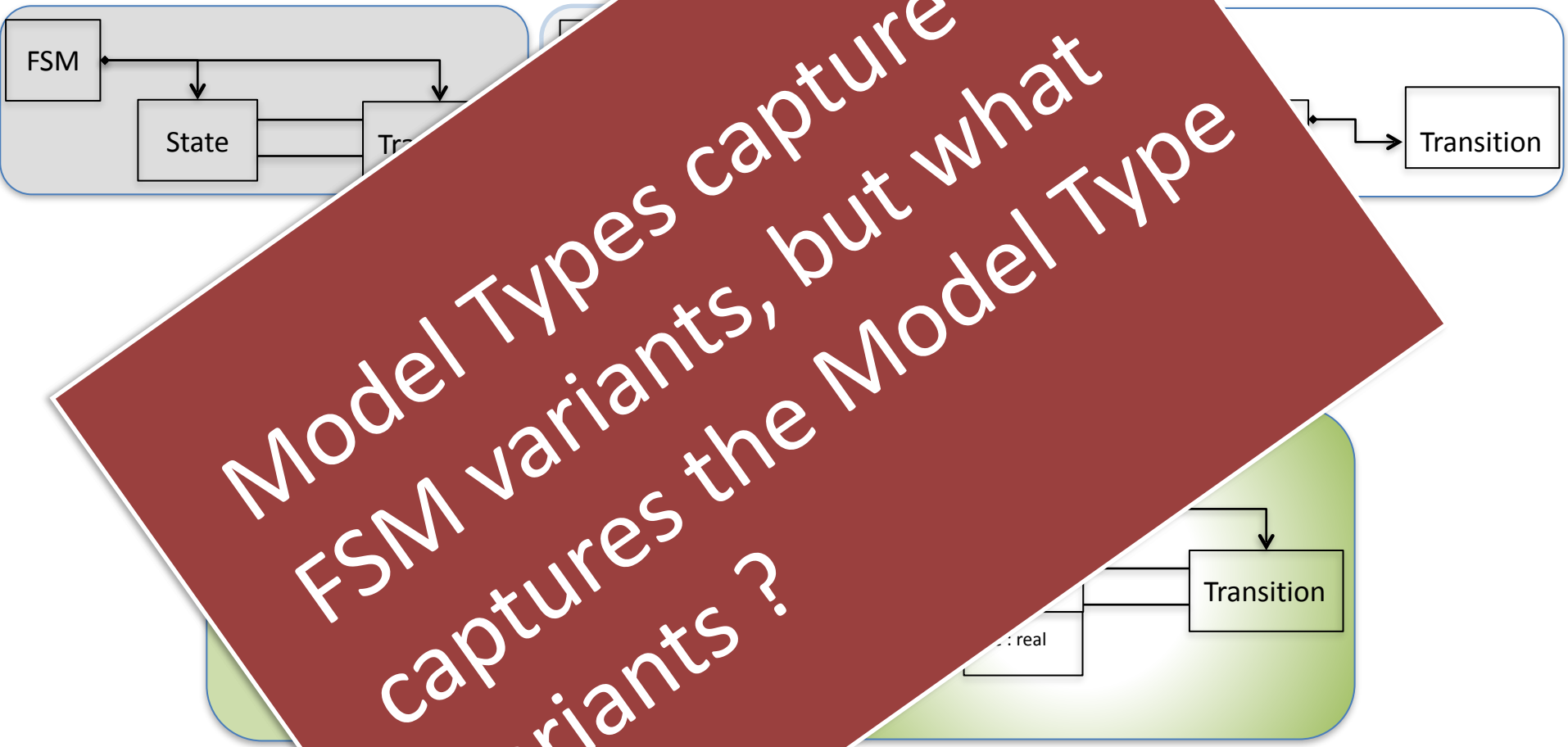
`accept() : boolean`



Applicable transformations:

`wcet() : real`

Variations o SM



Model Types capture the FSM variants, but what captures the Model Type Variants ?

Applica

accept

Applicable transformations:

`wcet() : real`

FMT: What's in a name?

***Intent:* Do not reinvent the wheel !**

Reuse existing techniques as much as possible!

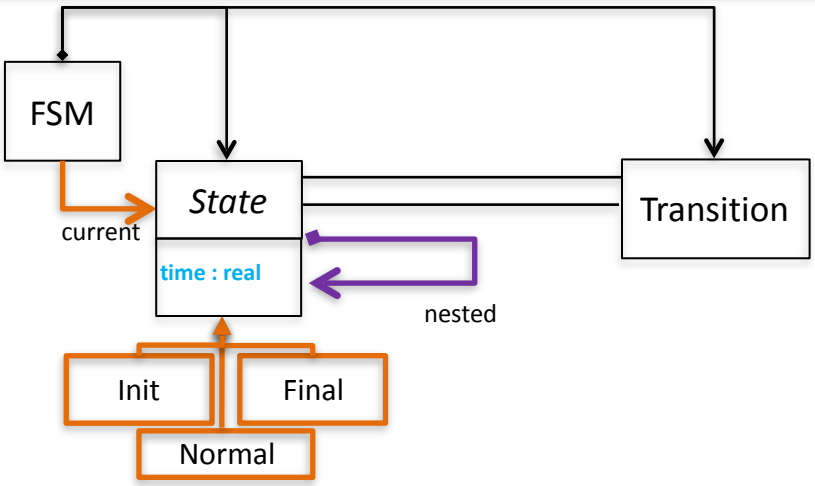
Manage explicitly your language assets

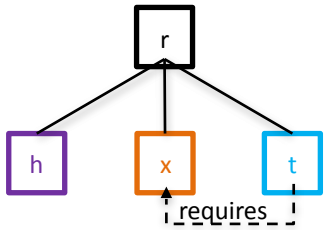
Operate with multi-granularity (both coarse-/fine-grained);

Create repositories of specialised domain assets

Configure your language !

FSM_FMT






```
r           / minimise() : FSM
r ∧ h       / flatten() : HFSM
r ∧ t ∧ x   / wcet() : real
r ∧ x       / accept() : Boolean
```

Domain Engineering Activities

Building FMTs

Building FMTs

Purpose

Define FMT construction approaches:

"Big Bang": Design FMTs explicitly w.r.t SPL paradigm

"Incremental": Start from a MT and incrementally add features and MM elements

Building FMTs

Purpose

Define FMT construction approaches:

"Big Bang": Design FMTs explicitly w.r.t SPL paradigm

"Incremental": Start from a MT and incrementally add features and MM elements

Building FMTs

Purpose

Define FMT construction approaches:

"Big Bang": Design FMTs explicitly w.r.t SPL paradigm

"Incremental": Start from a MT and incrementally add features and MM elements

Challenges: providing construction primitives that support merging similar elements and features, correct by construction FMTs, evolution…

Validate FMTs

Validate FMTs

Purpose

Validate FMTs

Purpose

Dealing with FMTs inconsistencies

Structural: conflicting Names, references/ multiplicities mismatches => Can be addressed with variability-aware type checking

Semantic: unintended interactions amongst transformations, transformations not meant to work on hierarchies, …

Can be addressed via SPL testing or verification

Validate FMTs

Purpose

Dealing with FMTs inconsistencies

Structural: conflicting Names, references/ multiplicities mismatches => Can be addressed with variability-aware type checking

Semantic: unintended interactions amongst transformations, transformations not meant to work on hierarchies, …

Can be addressed via SPL testing or verification

Challenges: Scalability of analyses, “verifiability” of transformations

Application Engineering

Configure and Derive an MT Product

Configure and Derive an MT Product

Purpose

Configure your DSML the same way you configure your car…

Configurator partially generated from the feature model

Product derivation techniques (e.g. pruning) to build desired MT automatically

Configure and Derive an MT Product

Purpose

Configure your DSML the same way you configure your car…

Configurator partially generated from the feature model

Product derivation techniques (e.g. pruning) to build desired MT automatically

Challenges

Partial configuration, user guidance on the relevance of elements (documentation issues)

Validate MTs

Validate MTs

Purpose

Validate MTs

Purpose

Perform QA activities that are too expensive at the domain engineering level (e.g. “integration” tests)

Validate MTs

Purpose

Perform QA activities that are too expensive at the domain engineering level (e.g. “integration” tests)

Validate MTs

Purpose

Perform QA activities that are too expensive at the domain engineering level (e.g. “integration” tests)

Challenges

Reusing validation artifacts from domain engineering, validating them (e.g. Mutation Analysis)

Matching and Customising MTS

Matching and Customising MTS

Purpose

Matching and Customising MTs

Purpose

Relating your MT with existing metamodels

Matching and Customising MTs

Purpose

Relating your MT with existing metamodels

Needed if your MT is not a DSML in itself but part of it

Matching and Customising MTs

Purpose

Relating your MT with existing metamodels

Needed if your MT is not a DSML in itself but part of it

For fully derived MTs, existing techniques apply

Matching and Customising MTs

Purpose

Relating your MT with existing metamodels

Needed if your MT is not a DSML in itself but part of it

For fully derived MTs, existing techniques apply

Matching and Customising MTs

Purpose

Relating your MT with existing metamodels

Needed if your MT is not a DSML in itself but part of it

For fully derived MTs, existing techniques apply

Challenges

Dealing with partial MTs => “variability-aware” matching

Conclusion

We proposed a vision leveraging Model Types and Feature Modelling to support product-line engineering of modelling languages

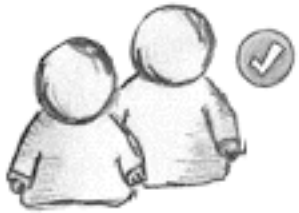
FMTs = support to manage reusable MM assets

Wishlist of high-level operations to work with FMTs

Future Work

Clafer

Lightweight Modeling Language



Designed with
usability in mind