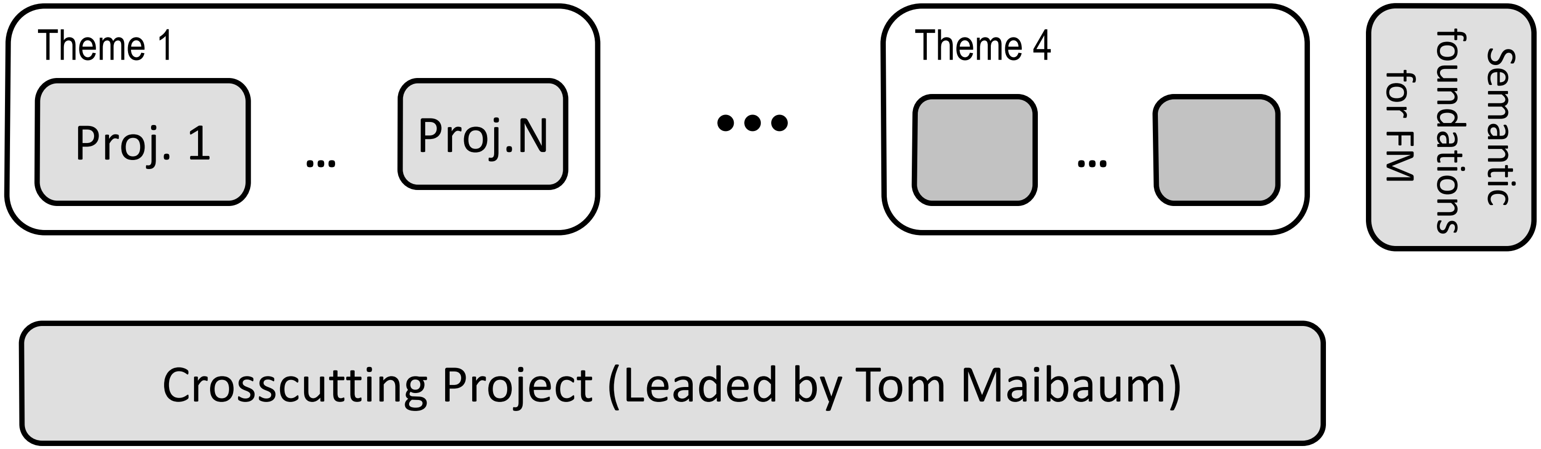
NECSIS: Tne Cross-Cutting Project

Zinovy Diskin
McMaster University, NECSIS
University of Waterloo

NECSIS Structure



Theme 1

Proj. 1

…

Proj.N

•••

Theme 4

…

Semantic foundations for FM

Crosscutting Project (Leaded by Tom Maibaum)

- Unified conceptual framework for model management (MMt)
- Unification of terminology and notation
- Common design and reasoning patterns

- To distill

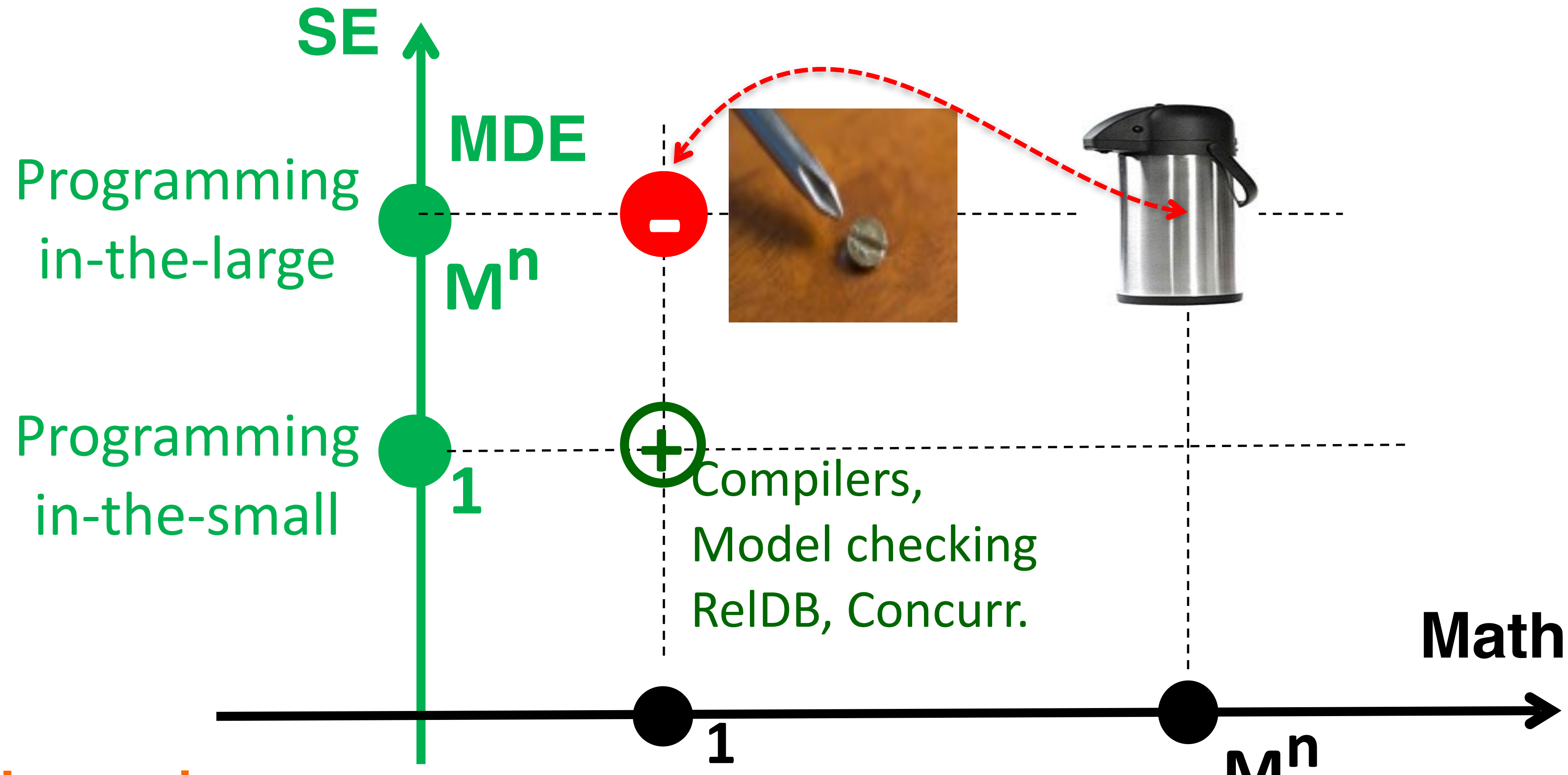
- To suggest

CC Results

- Usage of the results by the NECSIS participants :(
  - Unknown and unusual math based on mappings
  - Unknown terminology and notation
  - **No tool support**
  - **Tutorial are needed**
-

- A sound theory of MMt based on math
  - Classification of MMt tasks
  - Notation and terminology
- Spec/Structural design patterns
  - **Normal** vs. **radical** design
    - book *What Engineers Know and How they Know It, 1990* by Walter Vincenti
- Reasoning techniques (in progress)

Math for the modern SE



Classical math

Automata/Grammars
Modal logic, FOL

Category theory

Fibrations, Monads,
Limits/Colimits,
Hyperdoctrines

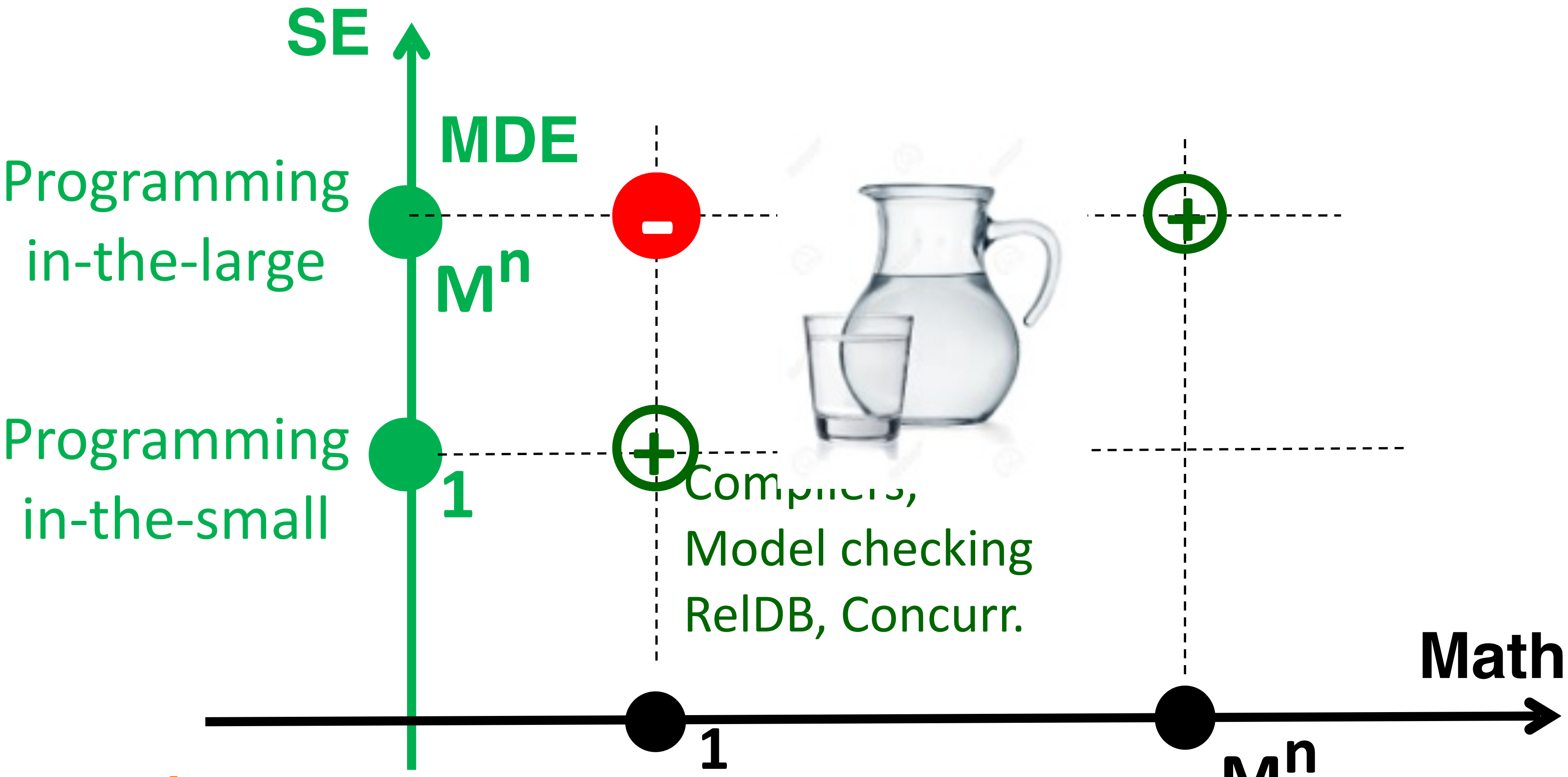
Legend:

1 – building **one** structure/ module is an issue

M – many: **composition** is the main issue

M^n – many of many

Math for the modern SE



SE

MDE

Programming in-the-large

M^n

Programming in-the-small

1

-

+

+

Compilers,
Model checking
RelDB, Concurr.

Math

1

M^n

Classical math

Automata/Grammars
Modal logic, FOL

Category theory

Fibrations, Monads,
Limits/Colimits,
Hyperdoctrines

Legend:
1 – building **one** structure/ module is an issue
M – many: **composition** is the main issue
M^n – many of many

Content

- Specification patterns for model management (40 min)
  - Model merge (Beh. modeling: choice) (15 min)
  - Model join/meet (Beh. modeling: concurrency) (5 min)
  - **Relational algebra for source-to-target MT (15 min)**
- Incremental BX and their taxonomy (0 min b/c of the upcoming NECSIS webinar on Mar 20)
- Foundations of feature modeling (8-10 min)*

*) Does not use category theory :)

Specification Patterns for Model Management

McMaster:
Hamid Gholizadeh,
Sahar Kokaly,
Tom Maibaum,
Zinovy Diskin

MDE adoption in industry

- The MDE idea is great but it may not fully fulfill its promise. Why?
- Tools may be a (big) issue.
- Why are tools not good?
- Jon Whittle’s Studies (published at **ICSE, Models**)
  - Width: 19 interviews with 19 MDE practitioners from 18 companies
  - Depth: 10 interviews with Eriksson AB + 10 interviews with Volvo Cars

Quotes from Whitlle's papers

- "We do not have a fine-grained way of knowing **which** MDE tools are appropriate **for which** jobs."
- "There is also a clear gap in the way that vendors **market** their tools and their real **capabilities**."
- *"And suddenly the tool doesn't do something expected and it's a nightmare for them."* [a direct quote from an interview]

Miscommunication

Whittle's Studies: some results

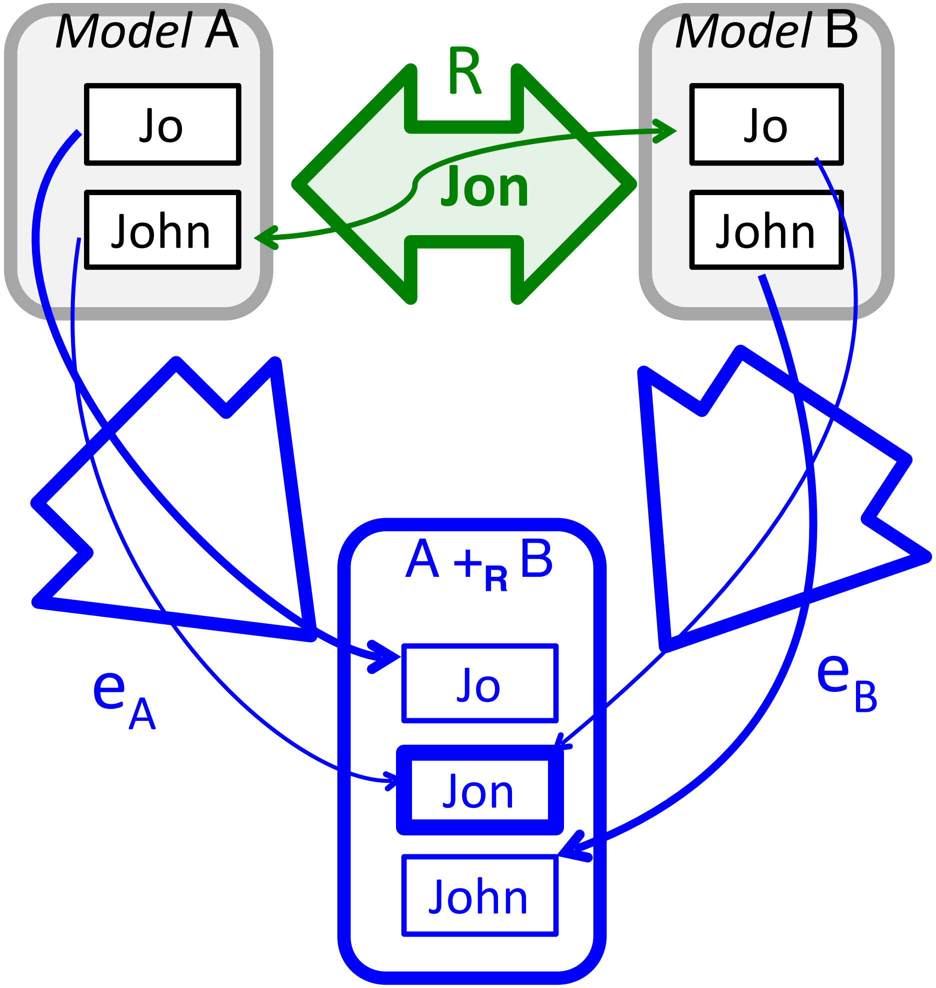
- Forty Issues preventing MDE adoption, and the miscommunication e:
  - – Technical defi DE tools (17/4),
  - – Inter (13/3),
  - – Ext (5/3),
  - – Soci rol) (5/2)
  - – Misco ounts to 25-50% per group
- “Define t eaning of words and you will avoid much discord” (René Descartes)

- Model merge
- Model sync
- Incremental model transf.
- Model refactoring
- .....

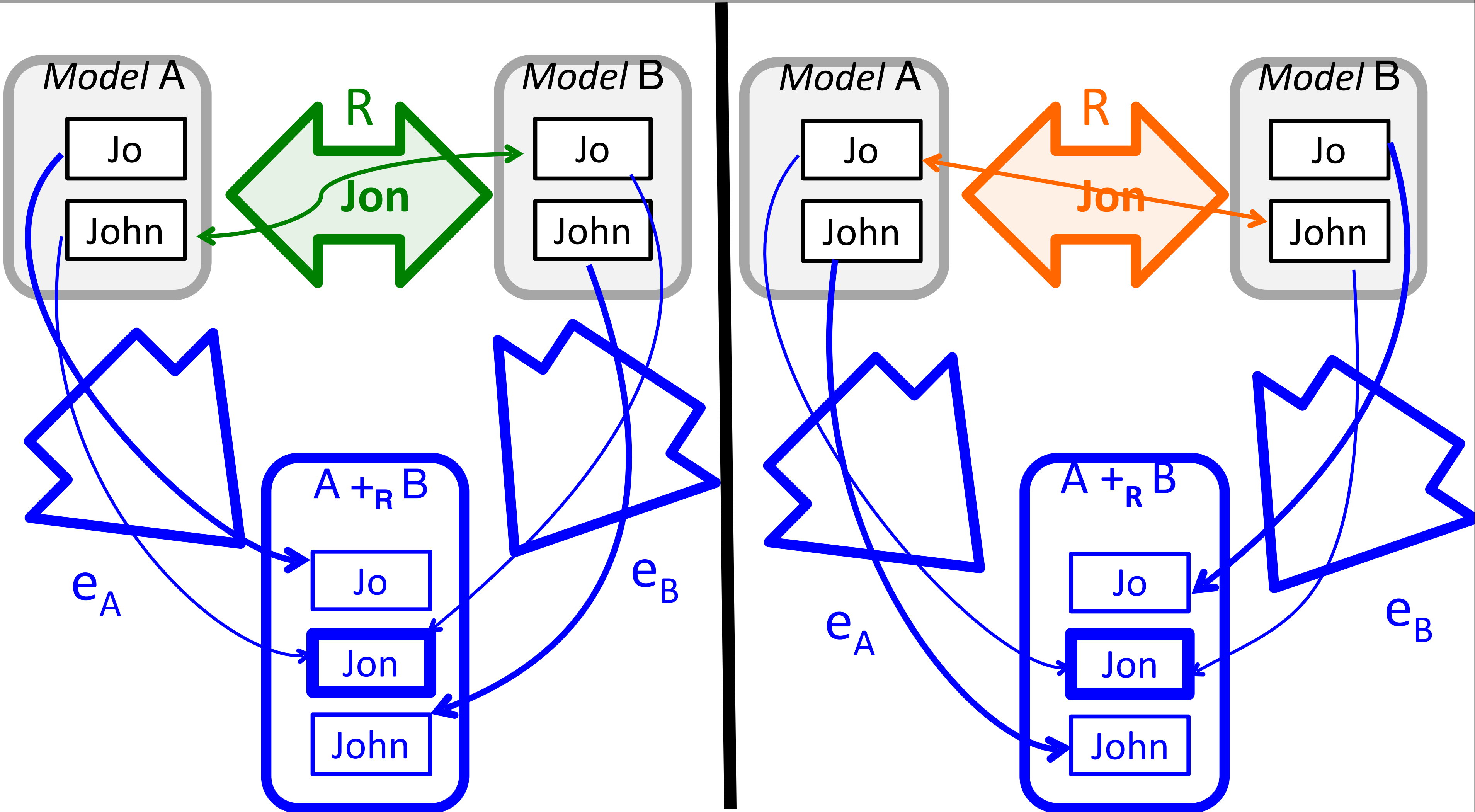
Specification patterns for MMt

- Intro
- Model merge via colimit
- Model join (meet, match) via limit
- Model translation via Cartesian monads :)
- Composing operations into workflows
- Summary

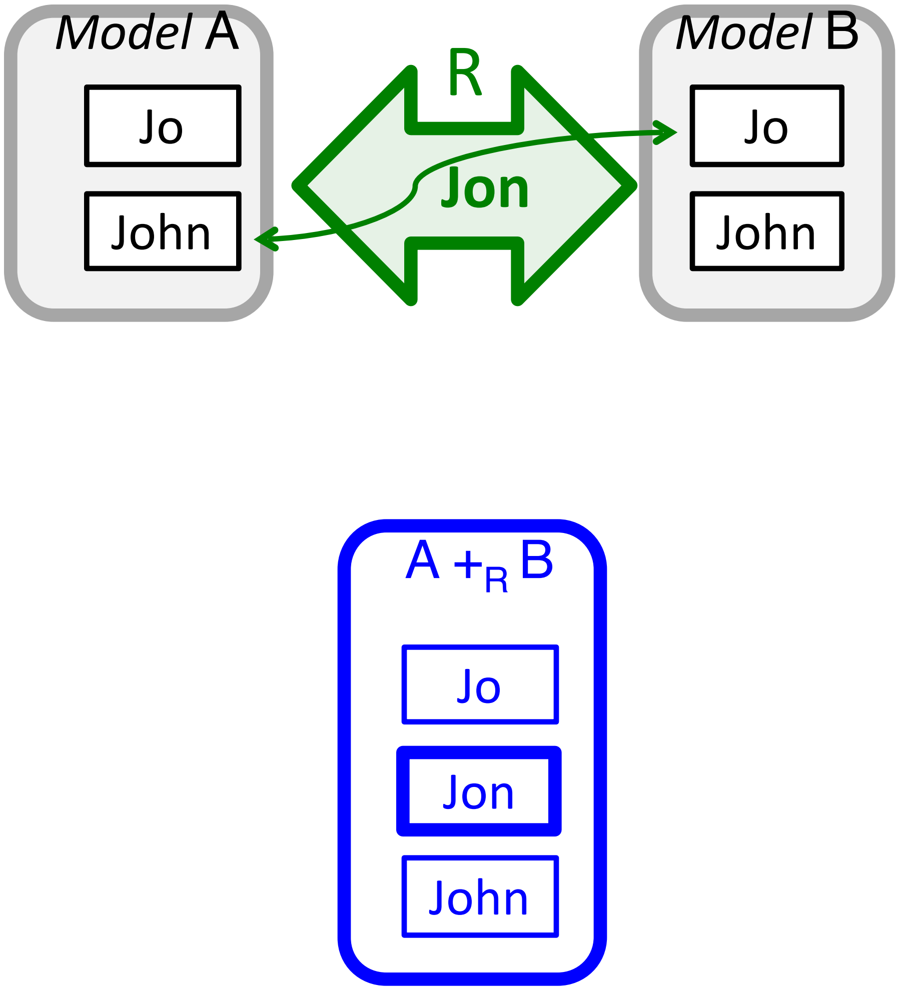
Model Merge after Match

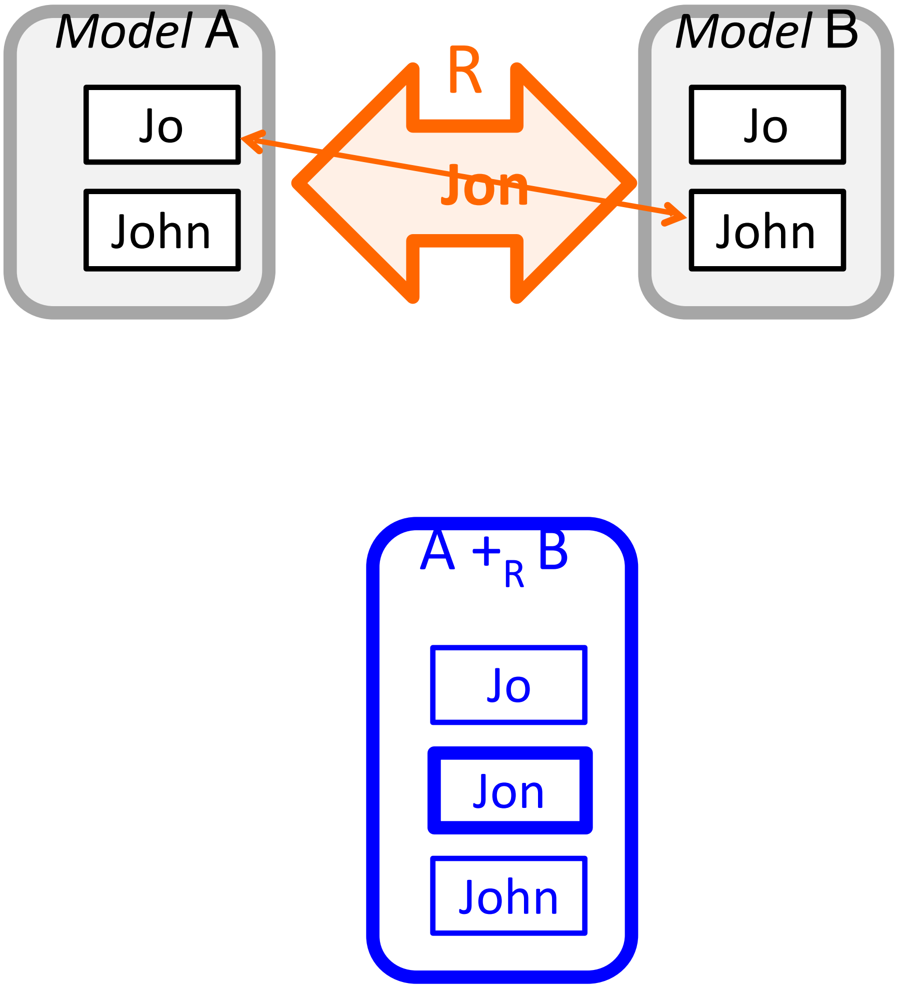


Model Merge with Green/Orange Match

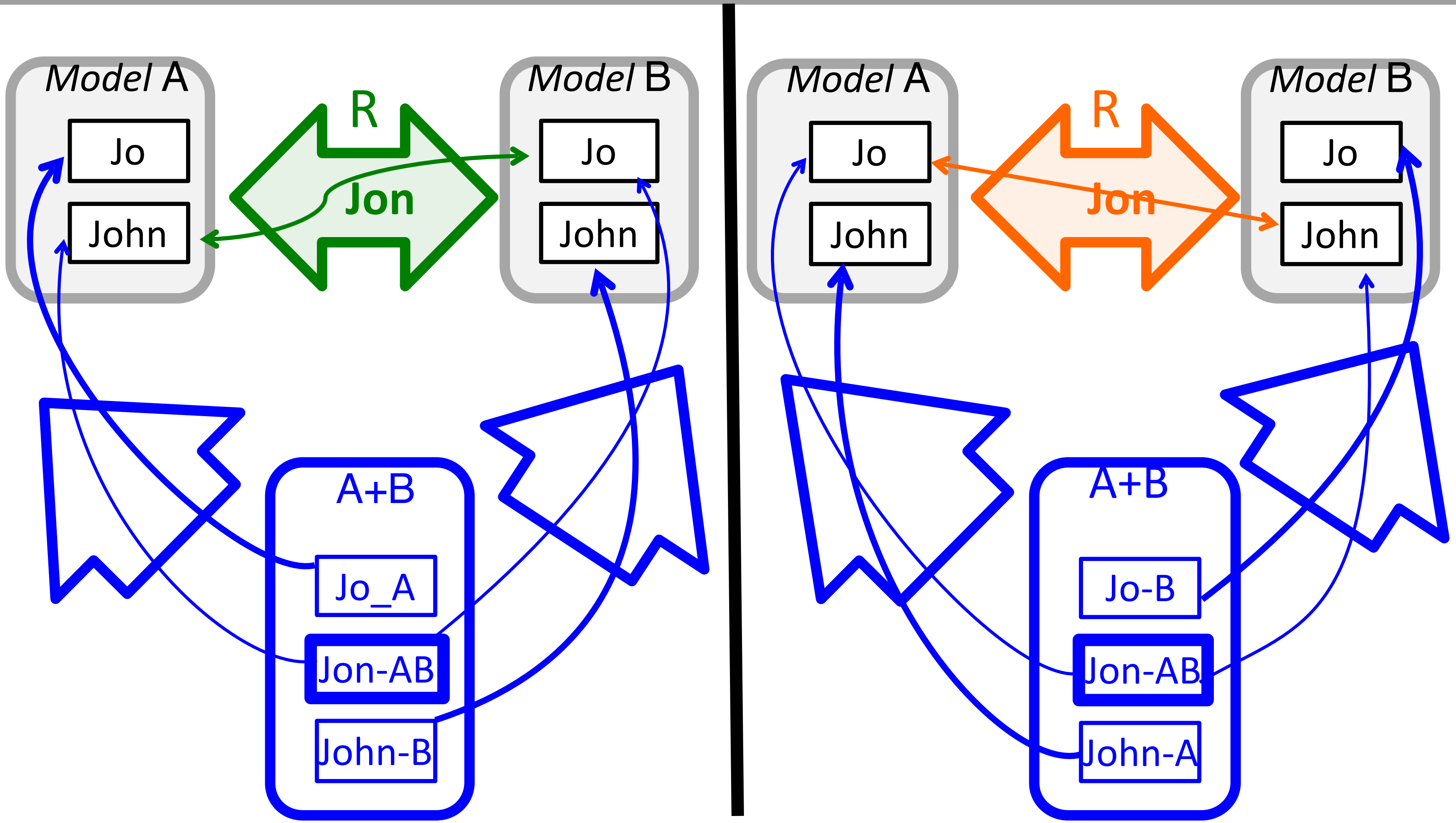


Merge without mappings

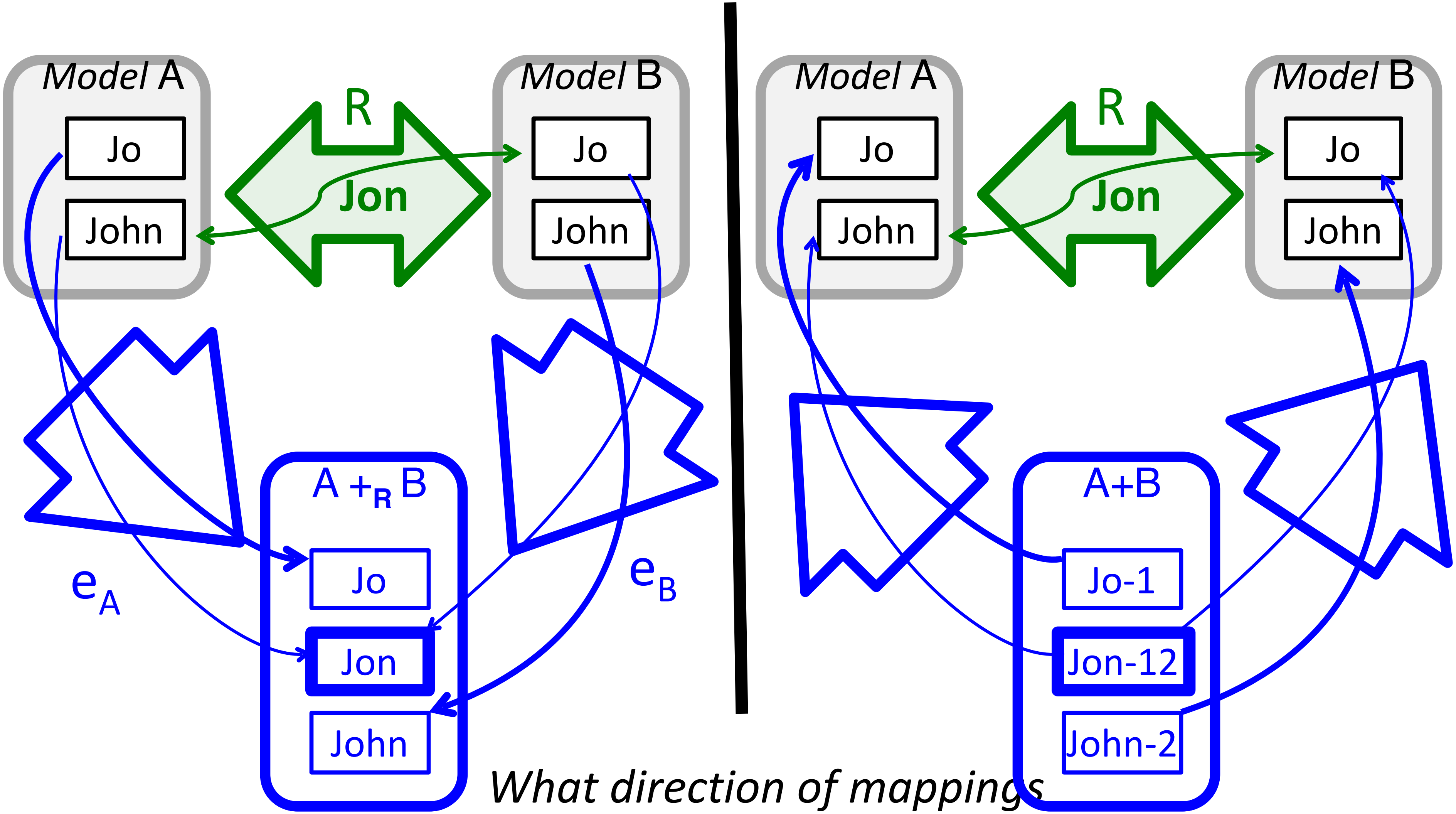




Merge with annotations

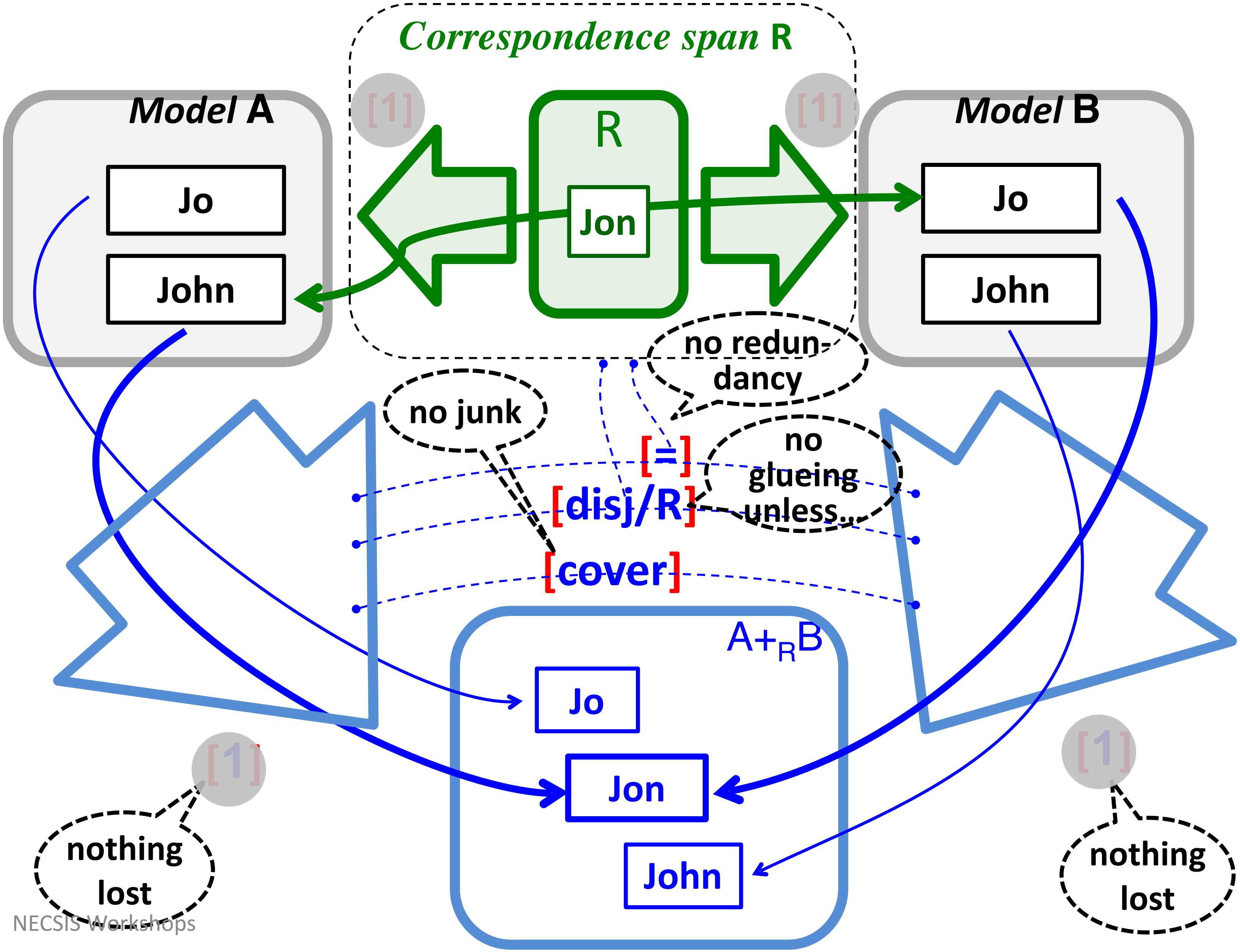


Merge via colimit vs. Merge with annotations

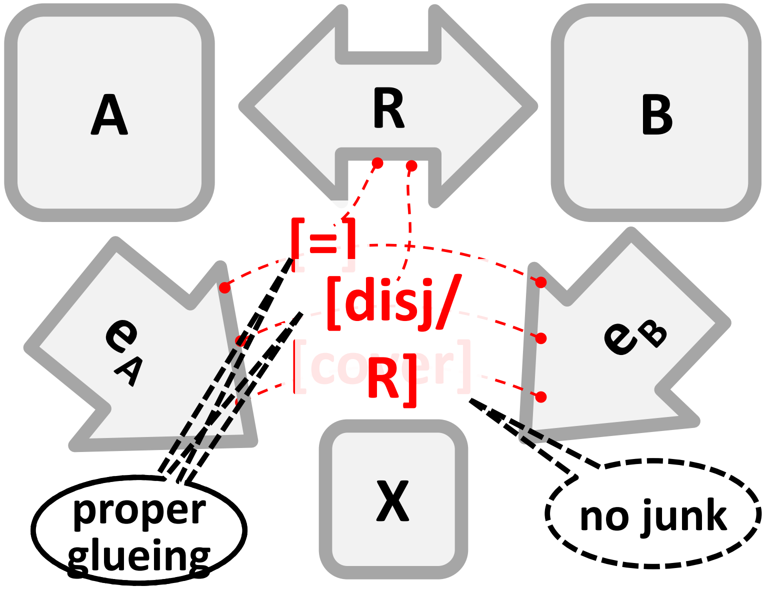


What direction of mappings e_A, e_B is "right"?

Merge via Colimit and Constraints



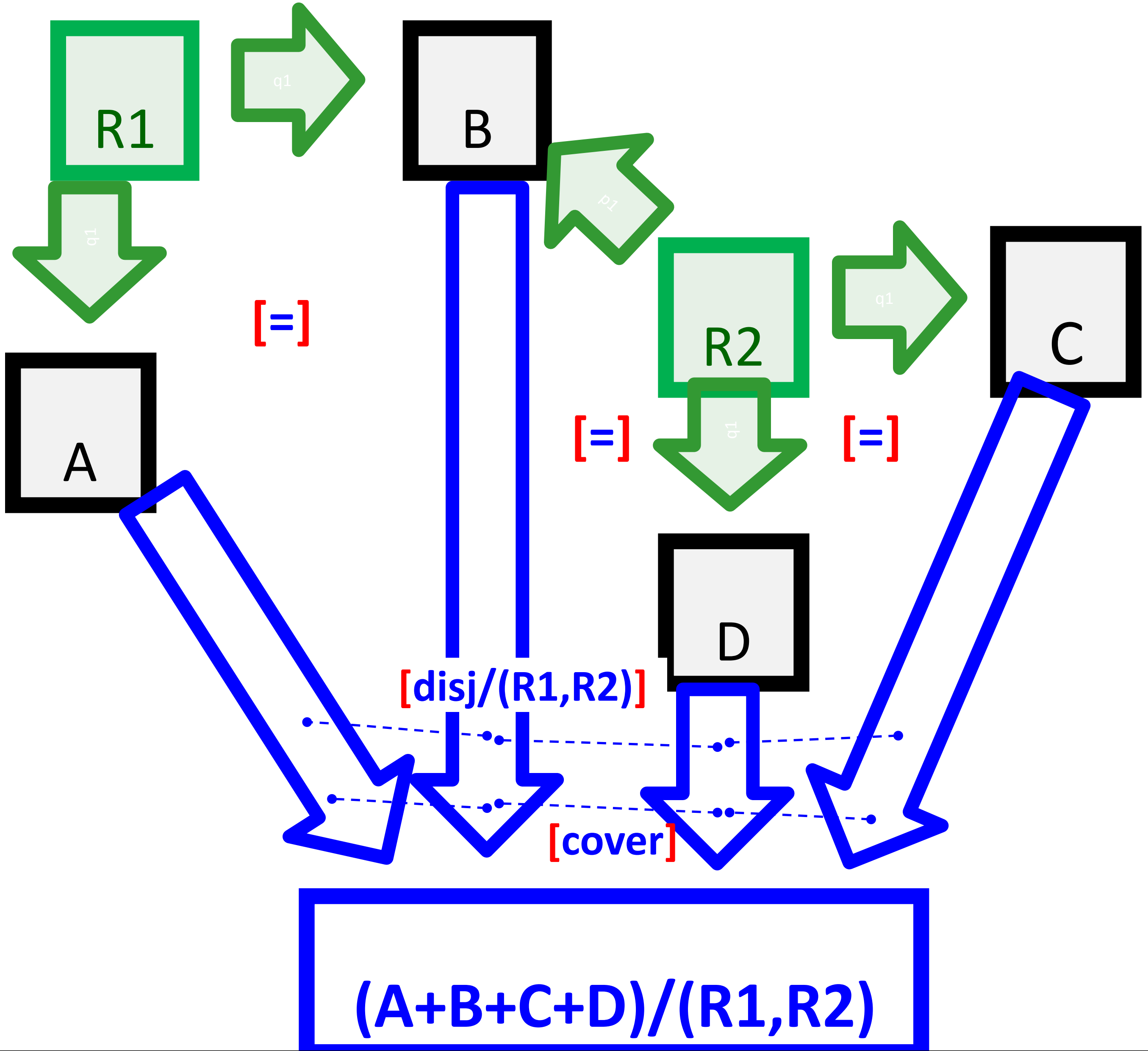
A great theorem of set merge



- **Theorem.** For any sets A,B and a corr. span R, there is one and only one (up to iso) set X together with maps e_A, e_B satisfying the three constraints.
- Hence, operation X = A $+_R$ B

- **Thesis (a la Church-Turing).** Any intuitive definition of set merge amounts to the formal operation A $+_R$ B.
- **$$$ Question:** Can the theorem, and the thesis, be generalized for richer structures: graphs, attributed graphs, Petri nets, models for a given metamodel?

N-ary merge



R1

q1

q1

B

p1

R2

q1

C

q1

[=]

A

[=]

[=]

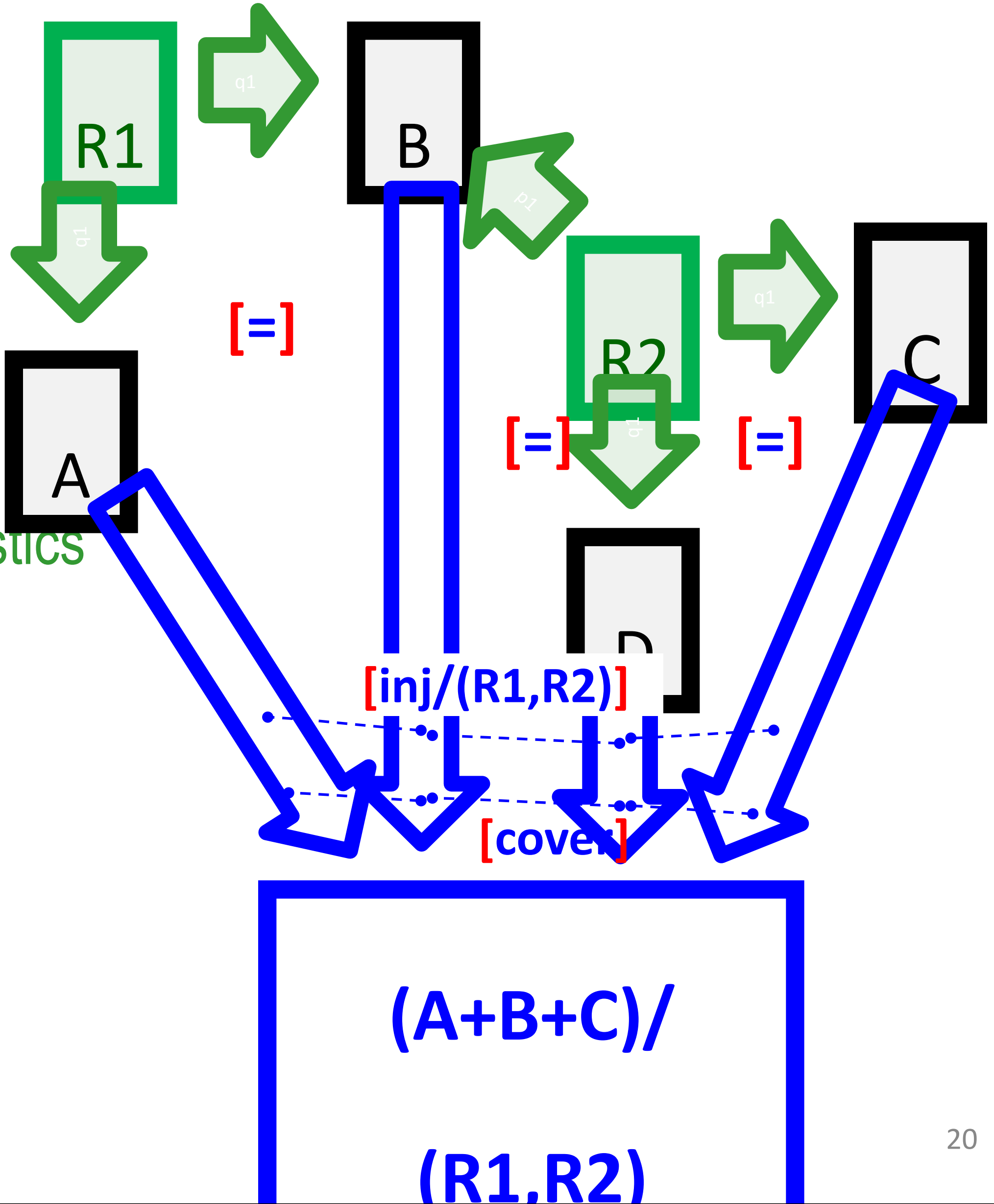
D

[disj/(R1,R2)]

[cover]

(A+B+C+D)/(R1,R2)

Four colors of model merge



Color Legend:

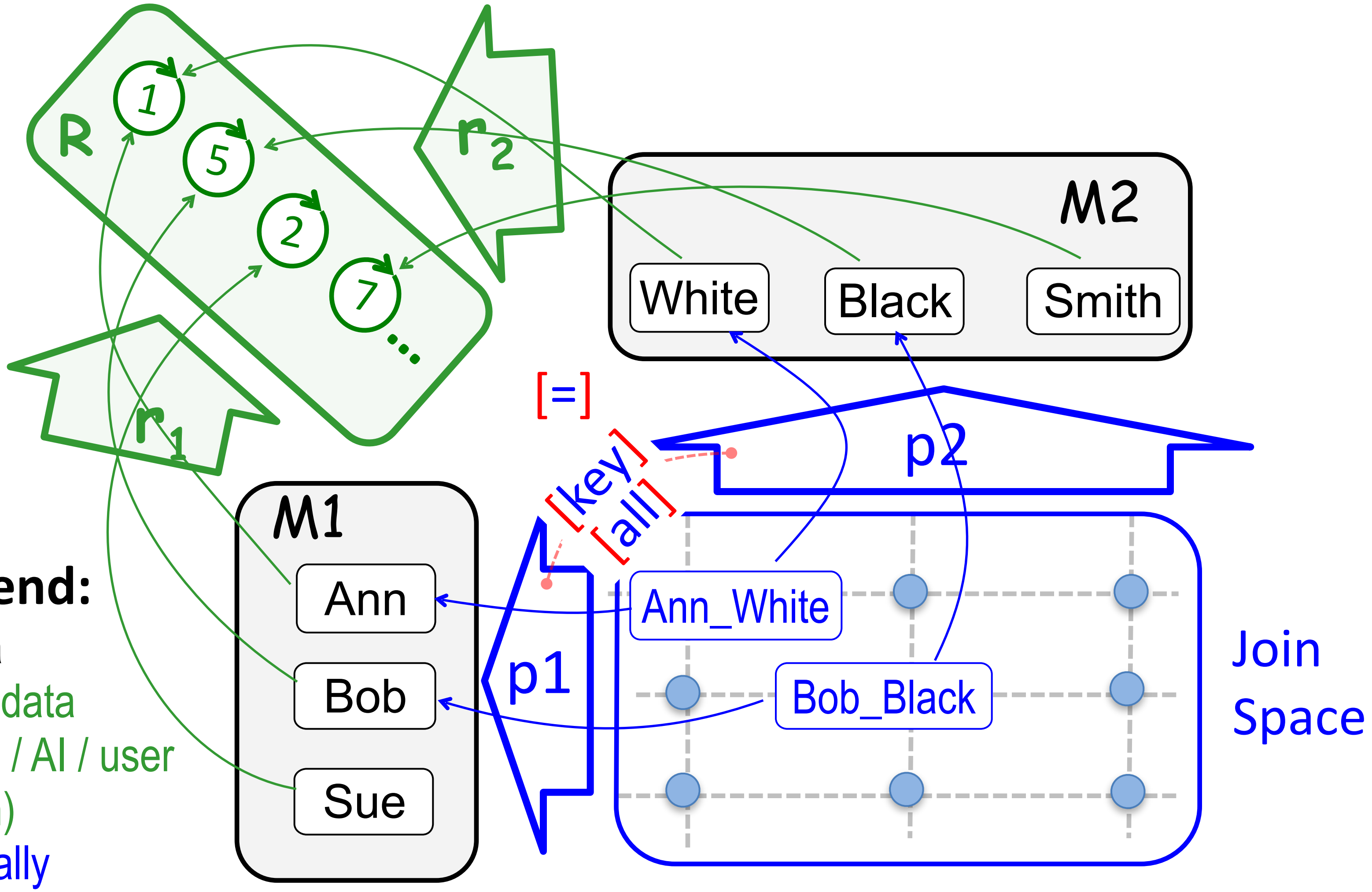
- ■ given data
- 🟩 model alignment/match (heuristics / AI / user interaction)
- 🟦 automatically computable

➢ mixing green and blue is bad

Specification patterns for MMt

- Intro
- Model merge (BM: choice) via colimit
- Model join (BM: concurrency) via limit
- Model translation via Cartesian monads :)
- Composing operations into workflows

Synchronizing sets: Example 1

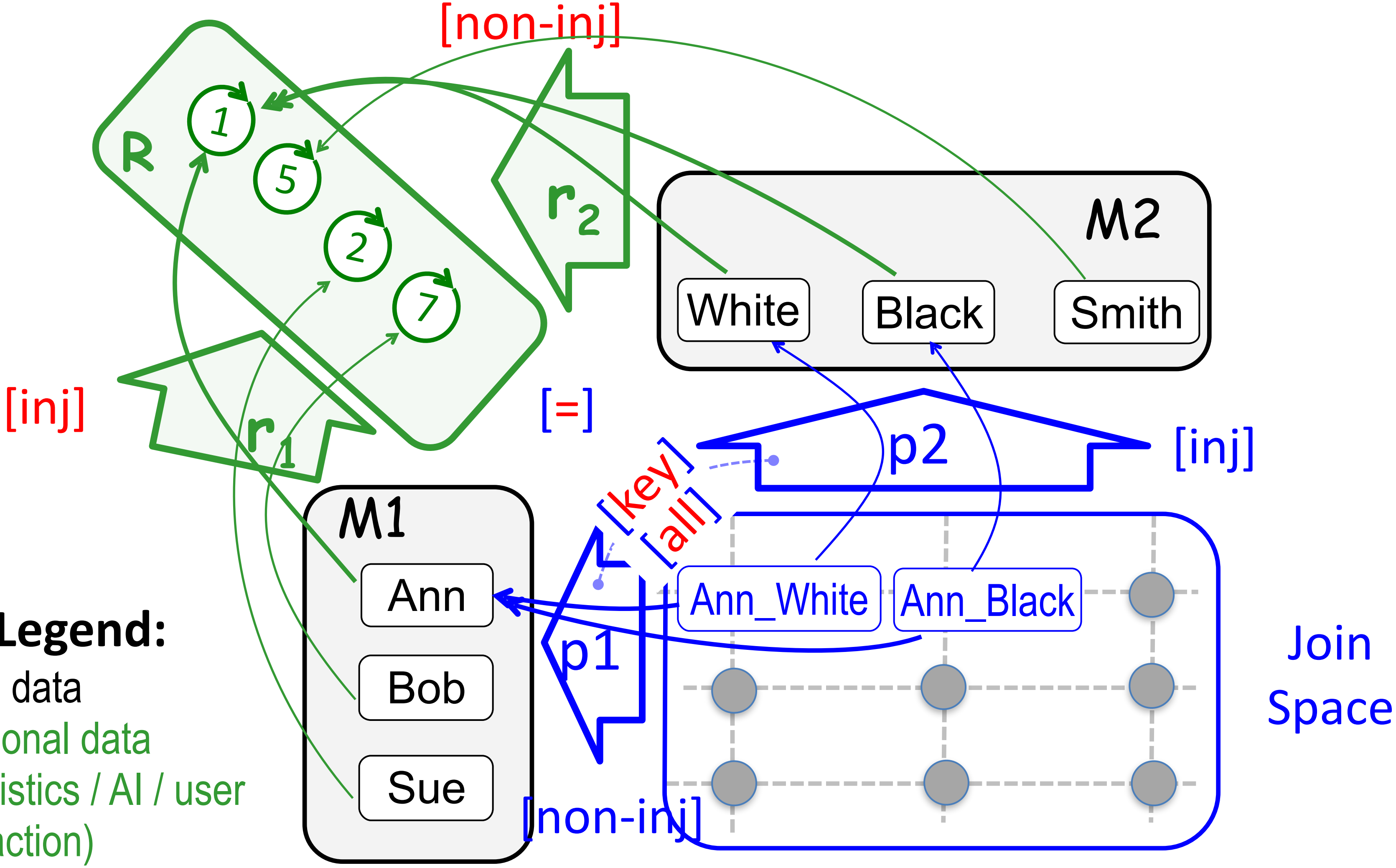


Color Legend:

- given data
- additional data (heuristics / AI / user interaction)
- automatically computable

$$M1\ x_R\ M2 = \{(n1@M1,\ n2@M2): r_1(n1) = r_2(n2)\}$$

Synchronizing sets: Example 2



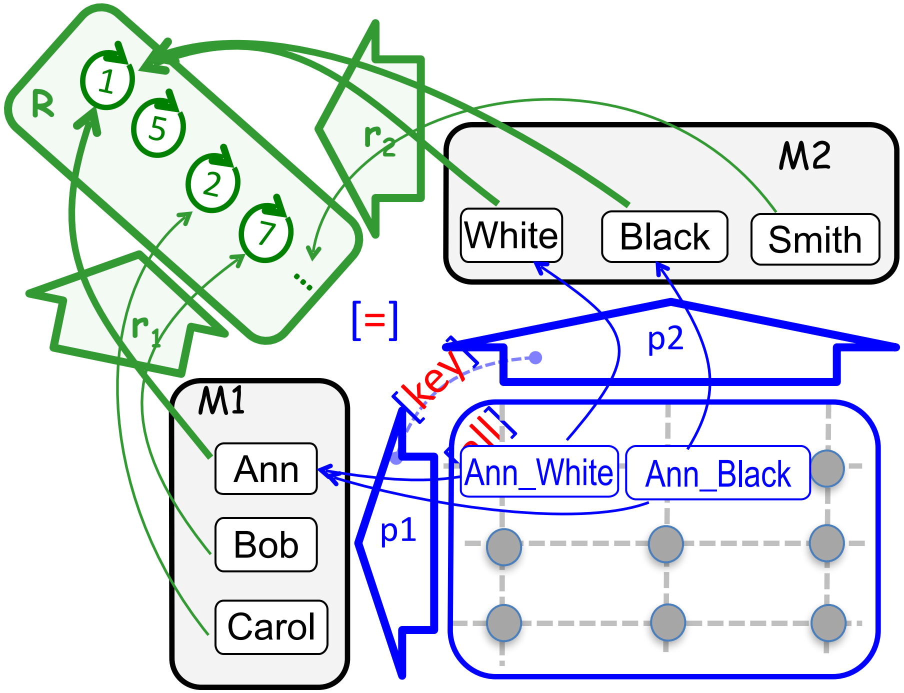
Color Legend:

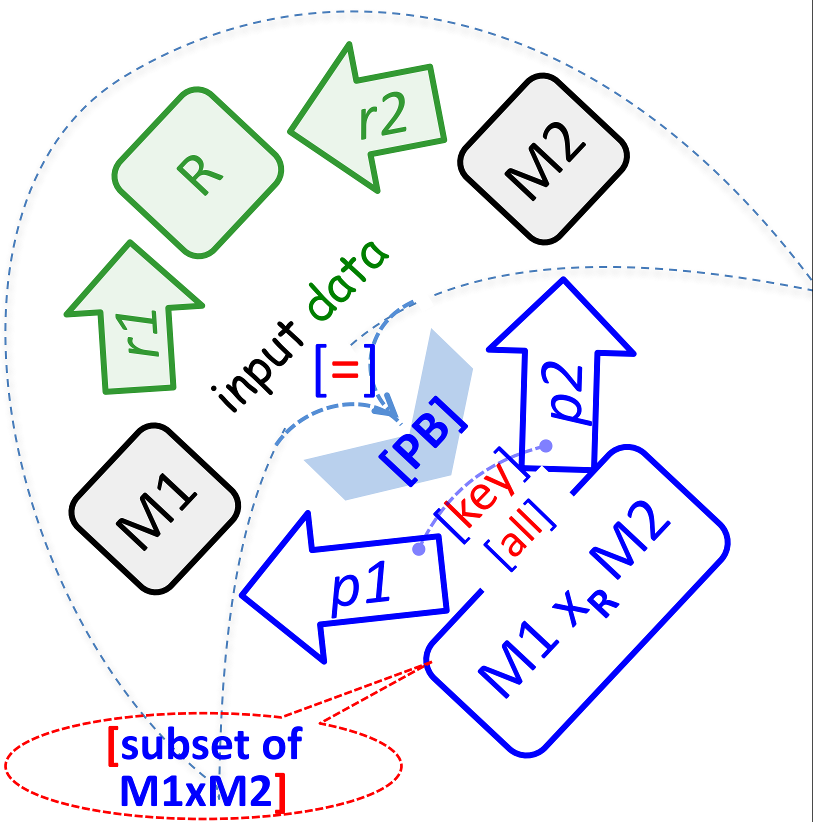
- given data
- additional data (heuristics / AI / user interaction)
- automatically computable

$$M1 \times_R M2 = \{(n1@M1, n2@M2): r_1(n1) = r_2(n2)\}$$

Abstraction: Synchronization as Pullback

Our concrete example → Categorical abstraction





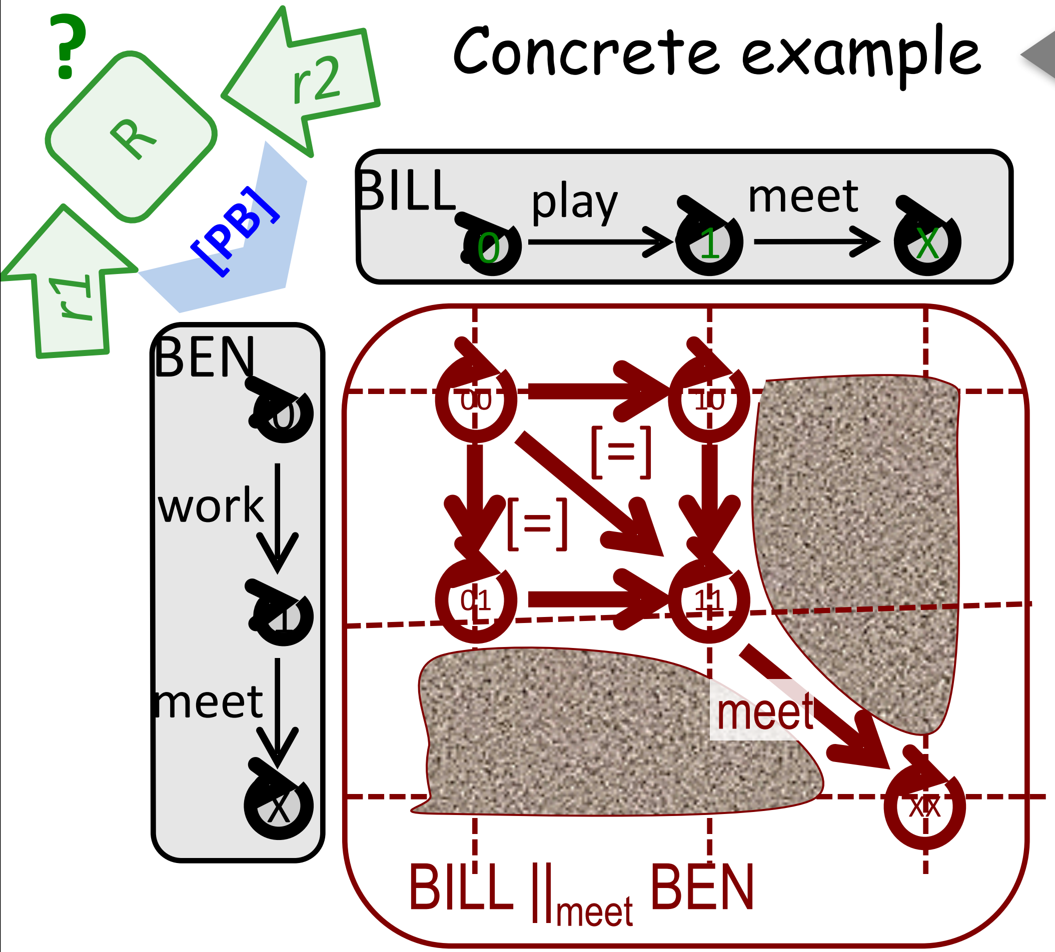
Color Legend:

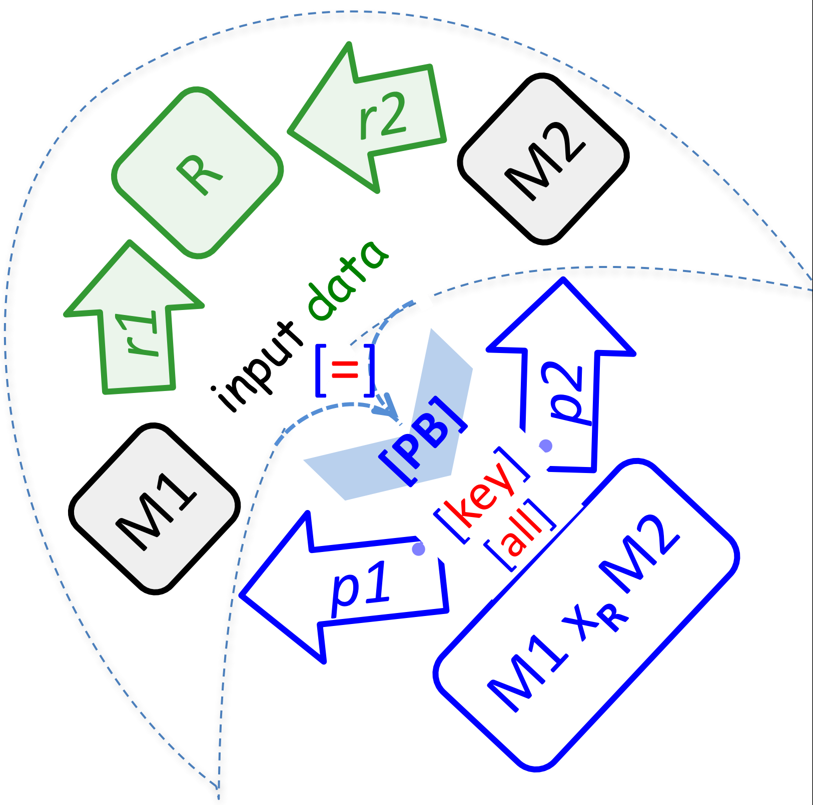
- ■ given data
- 🟩 additional data (heuristics / AI / user interaction required)
- 🟦 automatically computable

$$M1\ x_R\ M2 := \{(e1, e2): e1@M1,\ e2@M2,\ r1(e1) = r2(e2)\}$$

Instantiation: Parallel composition as Pullback

Concrete example ← Categorical pattern



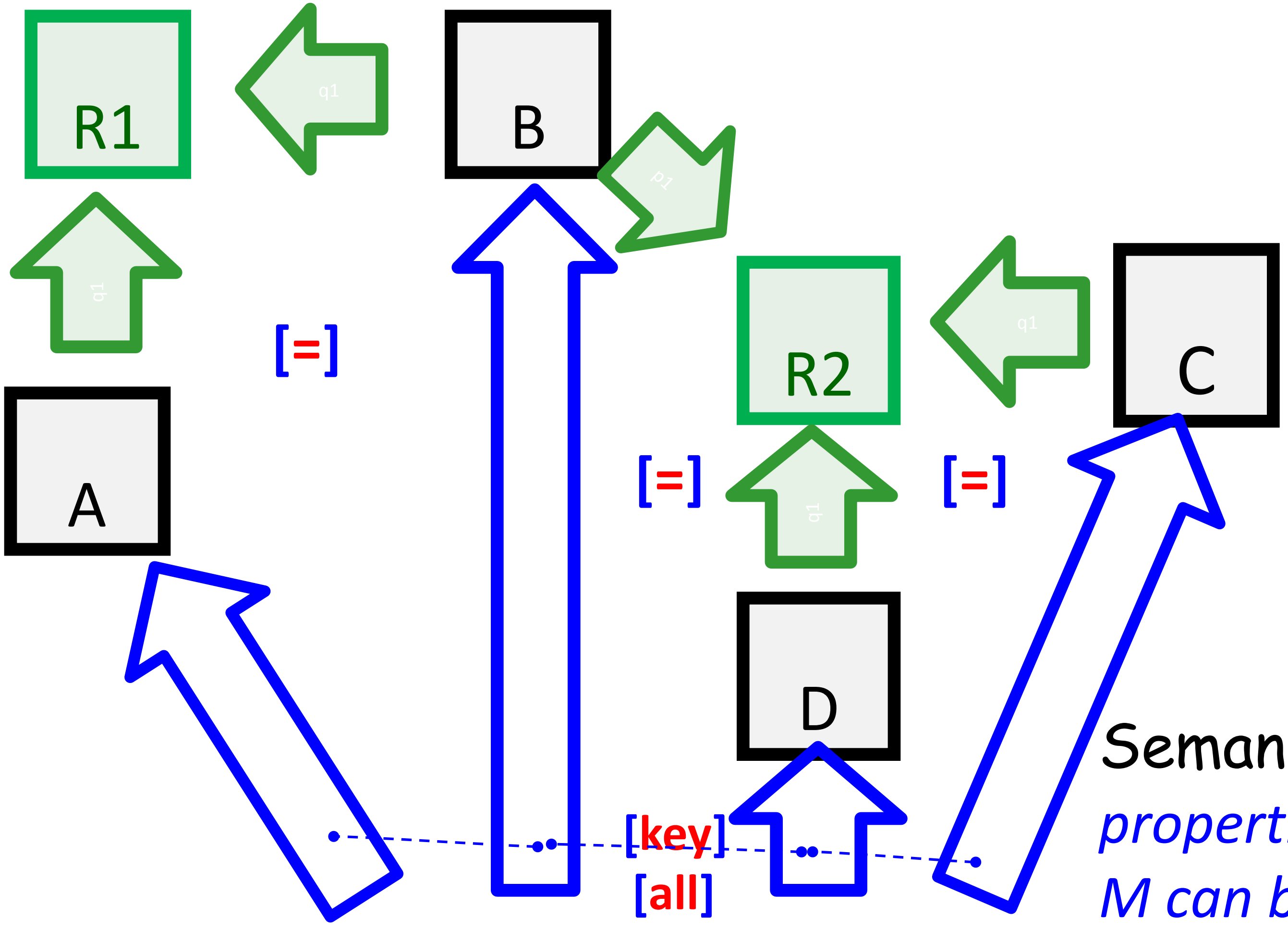


Color Legend:

- ■ given data
- ■ additional data (heuristics / AI / user interaction)
- ■ automatically computable

$$M1 \times_R M2 := \{(e1, e2) : e1@M1,\ e2@M2,\ r1(e1) = r2(e2)\}$$

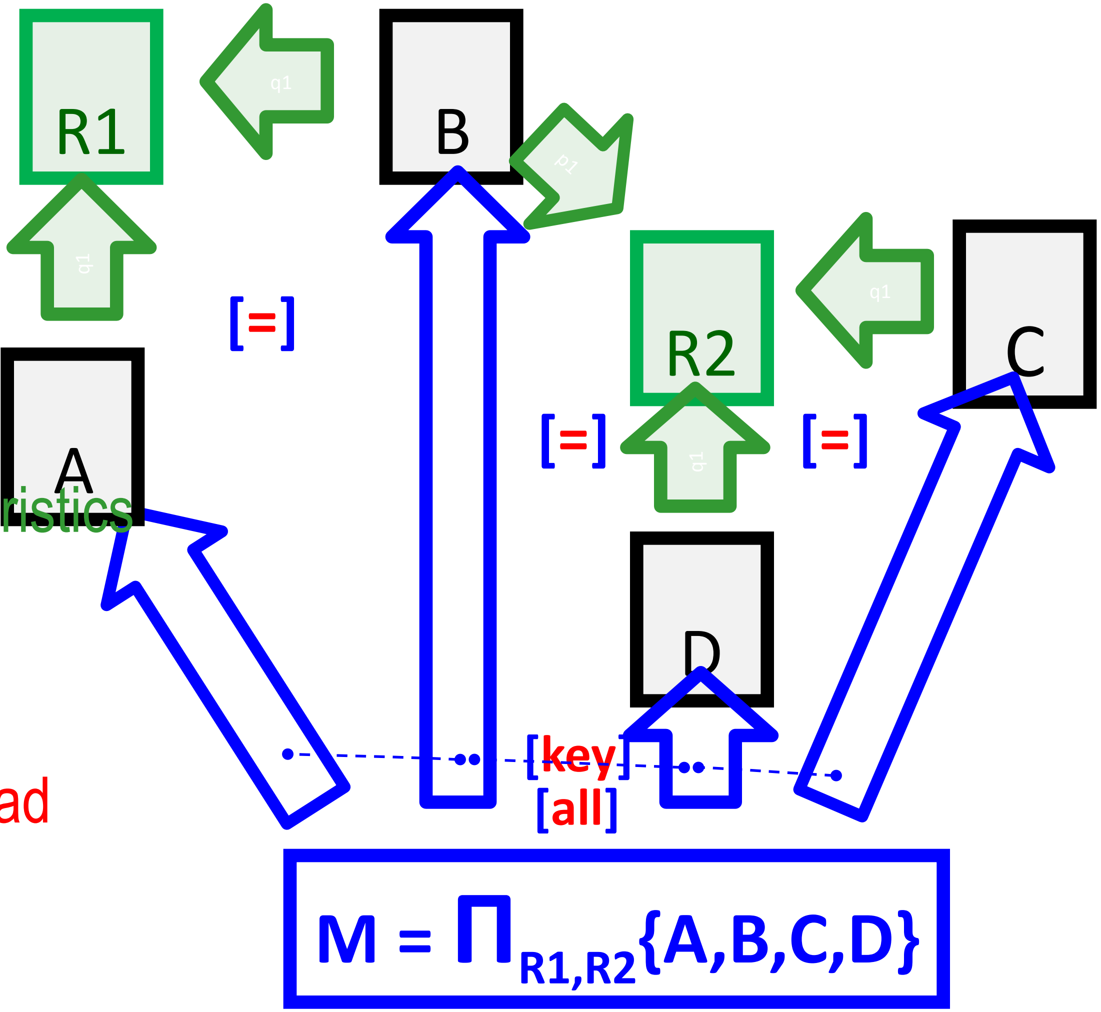
N-ary join



$$M = \Pi_{R1,R2}\{A,B,C,D\}$$

Semantics. *LTL/CTL properties of the process M can be derived from LTL/CTL properties of the components A,B,.. and their sync.*

Four colors of model join



$M = \Pi_{R1,R2}\{A,B,C,D\}$

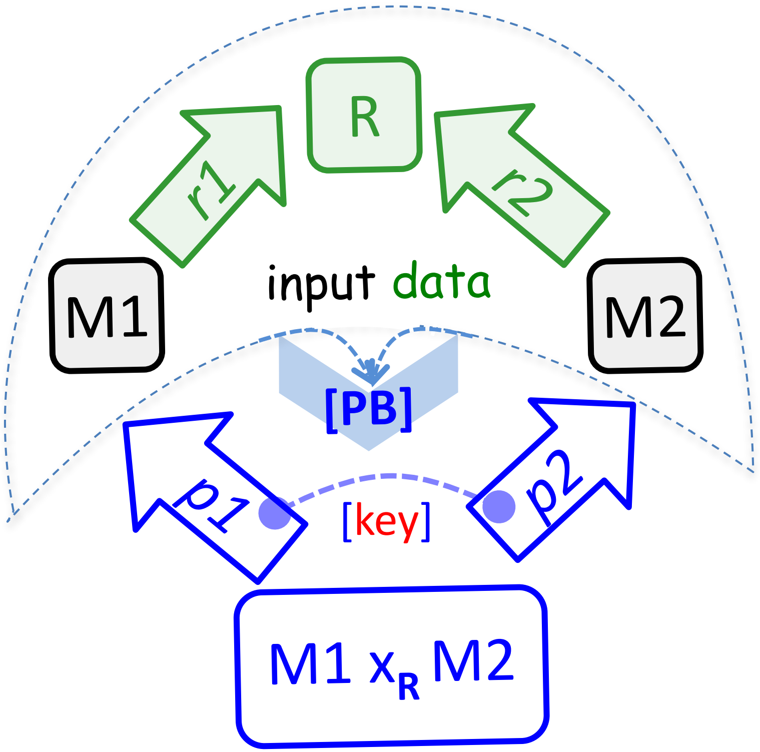
Color Legend:

- given data
- model alignment/match (heuristics / AI / user interaction)
- automatically computable

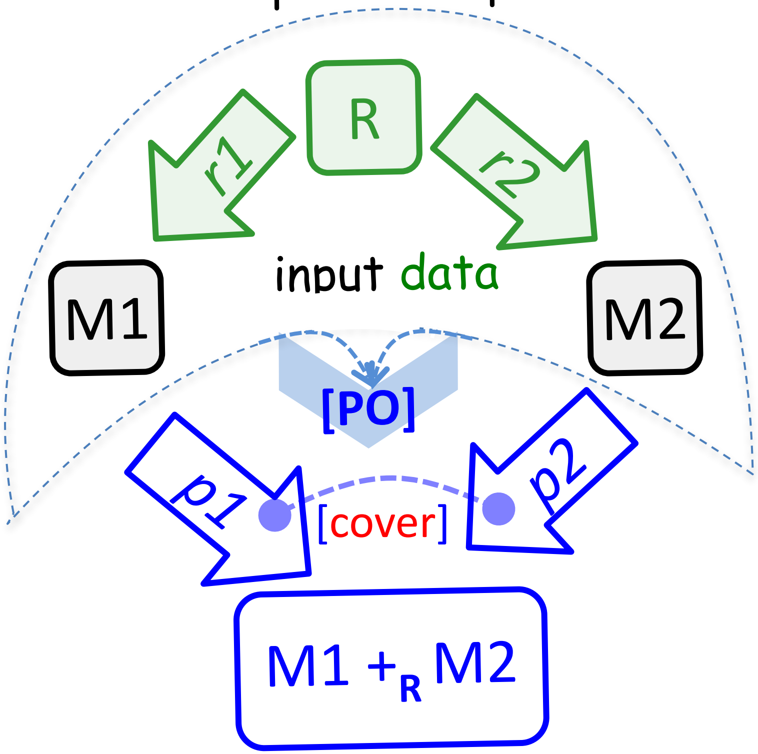
- mixing green and blue is bad

Duality of join and merge (Pullback vs. Pushout)

coSpan of maps



Span of maps



M1 x_R M2 := {(e1,e2) @ M1xM2:
r1(e1) = r2(e2) }

AND-composition/Concurr.
(limit)

M1 $+_R$ M2 := (M1 U M2) / R

OR-composition/Choice
(colimit)

Benefits of Merge & Join as Colimit (PO) & Limit (PB)

- Intelligent working with names
- Multi-ary complex merge & match are captured
- Separation of concerns (***Blue*** vs. ***Green***)
- Mathematical machinery to prove properties
  - PB is relational join. Hence, relational techniques can be applied
- Traceability mappings are always there

Specification patterns for MMt

- Intro
- Model merge (BM: choice) via colimit
- Model match (BM: concurrency) via limit
- Model translation via Cartesian monads :)
- Composing operations into workflows

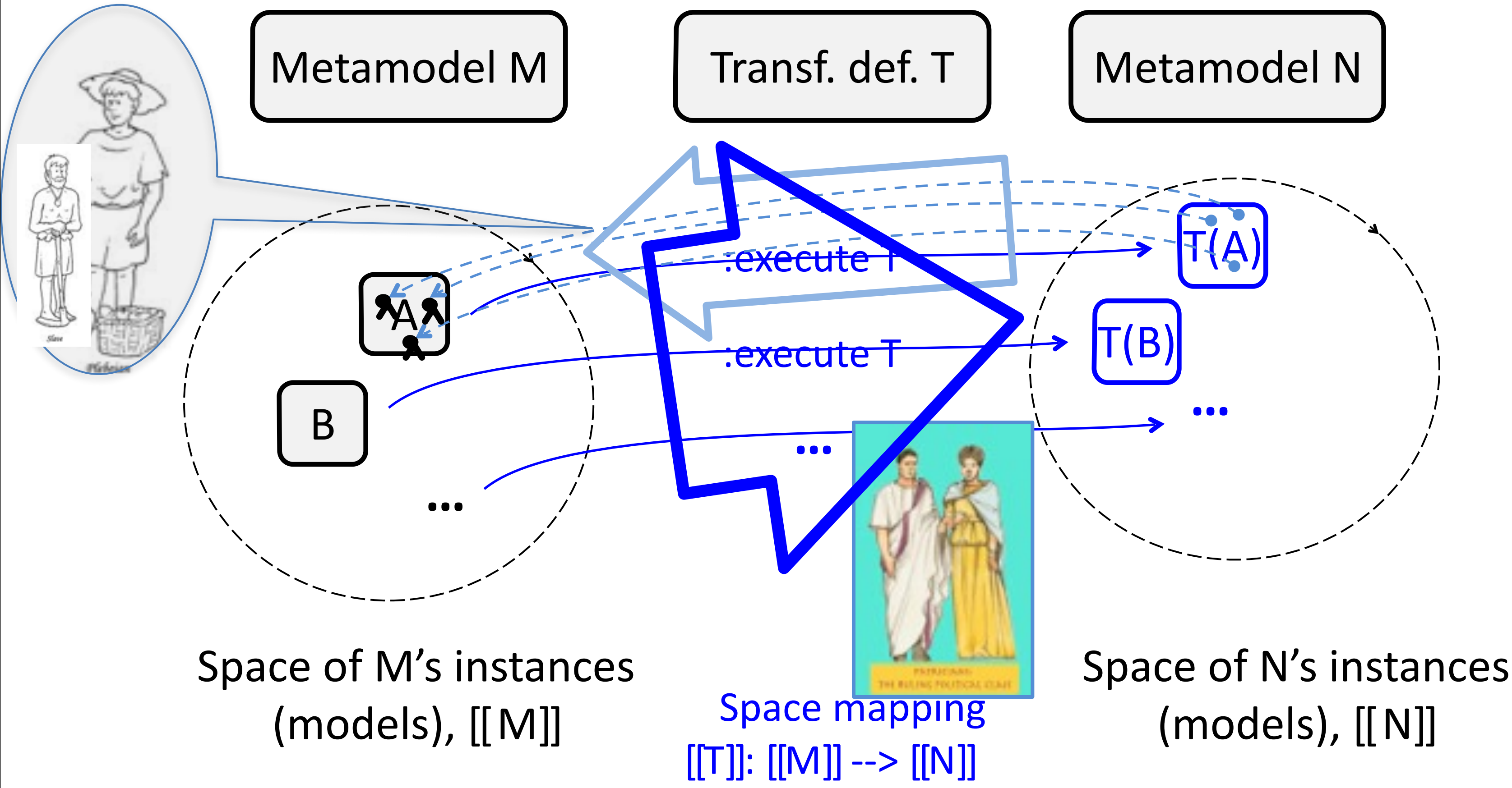
Towards Relational Algebra for Model Translations (just started)

McMaster:
Hamid Gholizadeh,
Sahar Kokaly,
Tom Maibaum

Waterloo:
Krzysztof Czarnecki,
Michal Antkiewicz,
Peiyuan Sun

Zinovy Diskin

Source-to-target model transf.



Metamodel M

Transf. def. T

Metamodel N

A

B

...

:execute T

:execute T

...

T(A)

T(B)

...

Space of M's instances (models), [[M]]

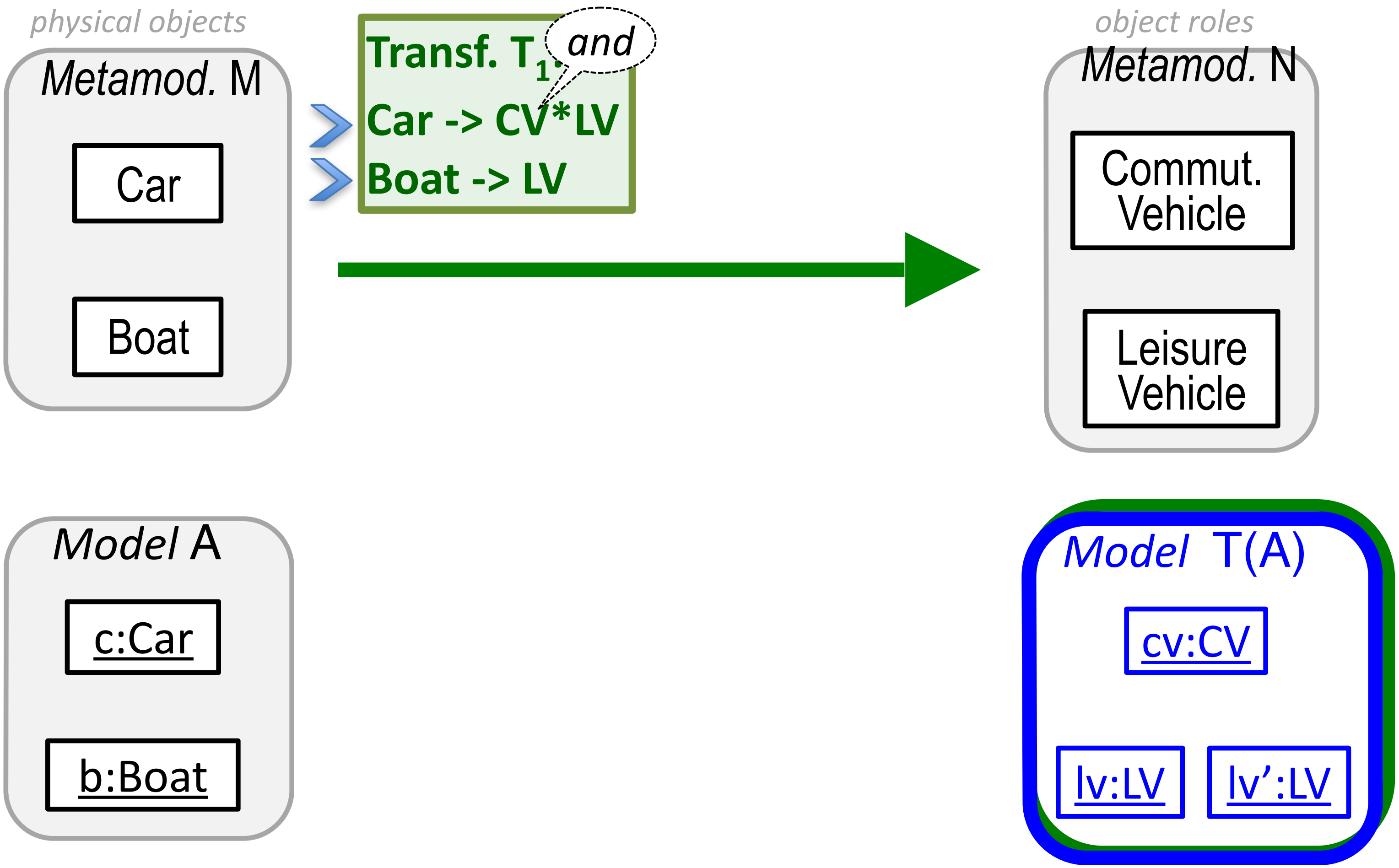
Space mapping [[T]]: [[M]] --> [[N]]

Space of N's instances (models), [[N]]

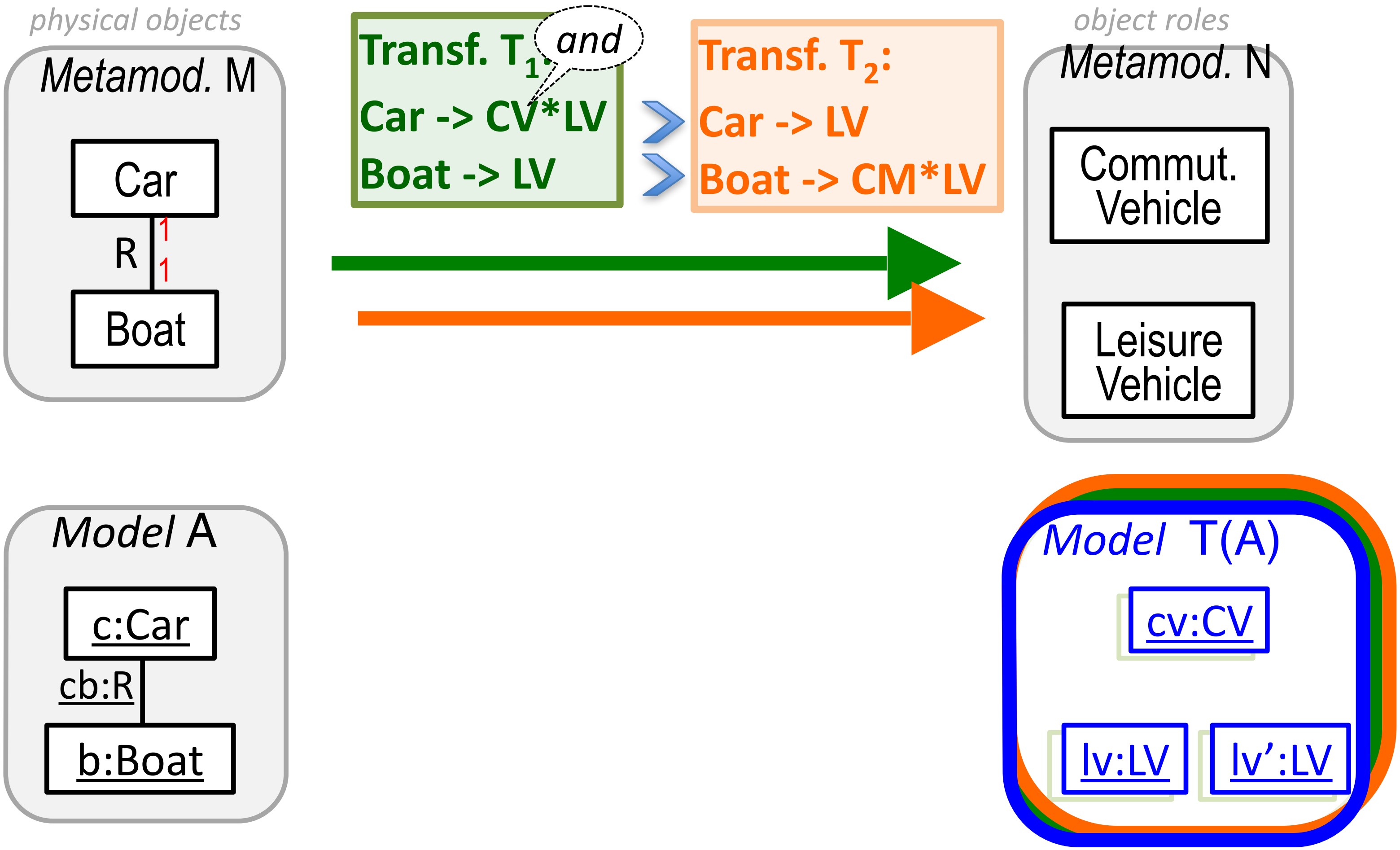
Color Legend:

- ■ given data
- ■ computed data

Model translation **w/out** traceability mappings



Model translation **w/out** traceability mappings



physical objects

Metamod. M

Car

R

1

1

Boat

Transf. T_1:

Car -> CV*LV

Boat -> LV

and

Transf. T_2:

Car -> LV

Boat -> CM*LV

object roles

Metamod. N

Commut. Vehicle

Leisure Vehicle

Model A

c:Car

cb:R

b:Boat

Model T(A)

cv:CV

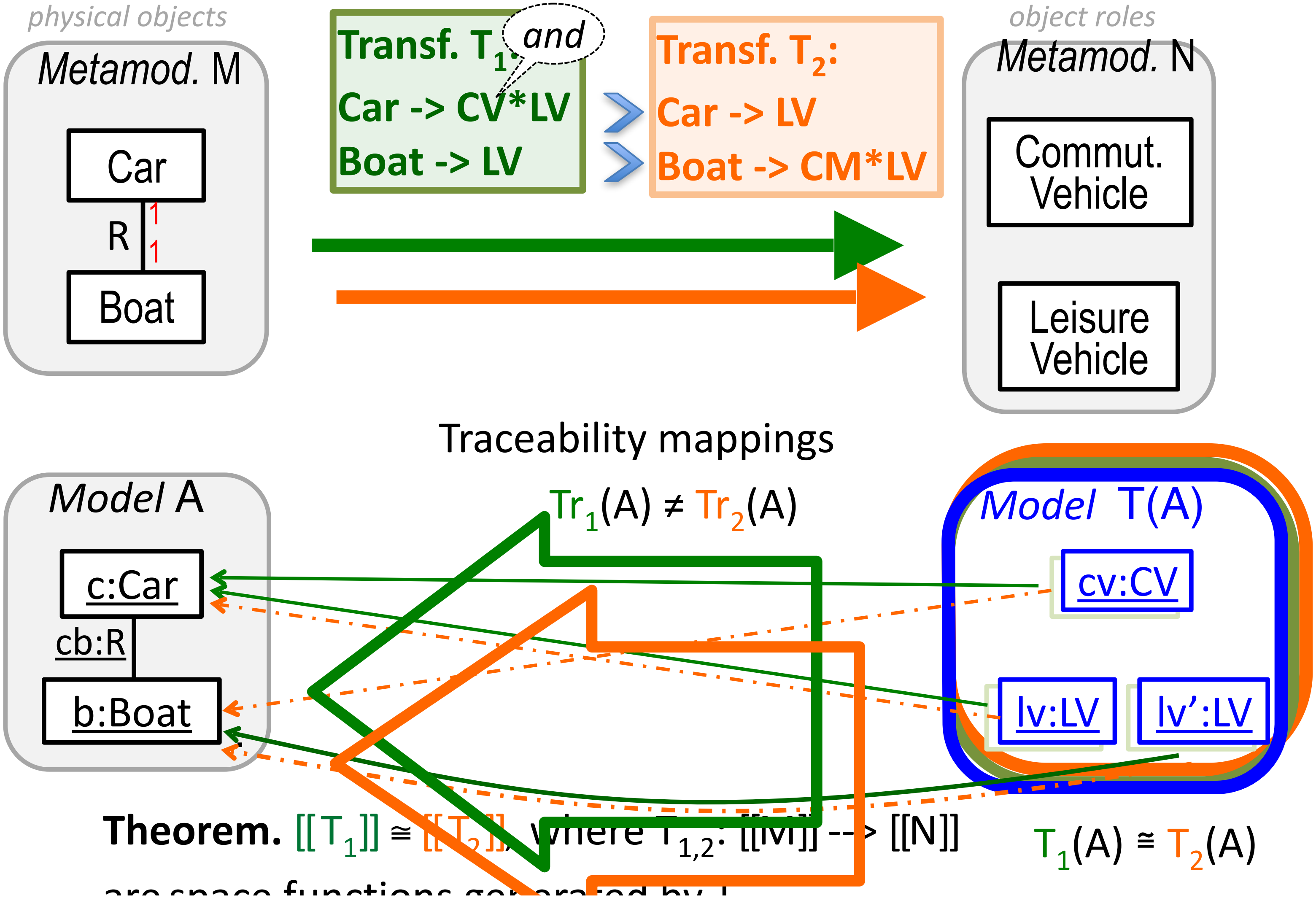
lv:LV

lv':LV

Theorem. $[[T_1]] \cong [[T_2]]$, where $[[T_{1,2}]]$: [[M]] --> [[N]] are space functions generated by $T_{1,2}$

$T_1(A) \cong T_2(A)$

Model translation **with** traceability mappings



physical objects

Metamod. M

Car

R

1

1

Boat

Transf. T_1:

Car -> CV*LV

Boat -> LV

and

Transf. T_2:

Car -> LV

Boat -> CM*LV

object roles

Metamod. N

Commut. Vehicle

Leisure Vehicle

Traceability mappings

$Tr_1(A) \neq Tr_2(A)$

Model A

c:Car

cb:R

b:Boat

Model T(A)

cv:CV

lv:LV

lv':LV

Theorem. $[[T_1]] \cong [[T_2]]$, where $T_{1,2}$: [[M]] --> [[N]]

are space functions generated by T

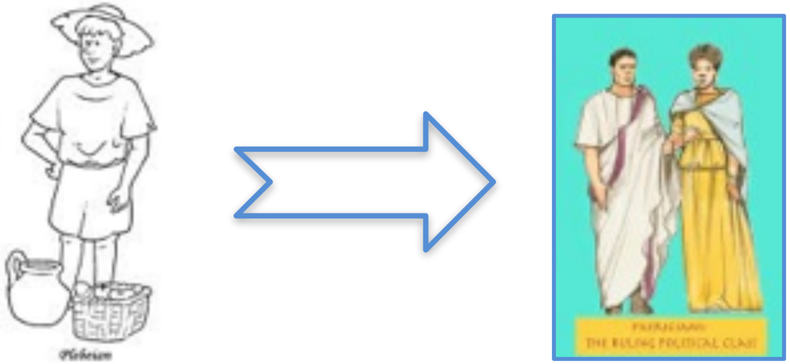
$T_1(A) \cong T_2(A)$

Theorem. $[[T_1]] \neq [[T_2]]$, where $T_{1,2}$: [[M]] --> ([[N]] **x Map([[N]] ,[[M]])**)

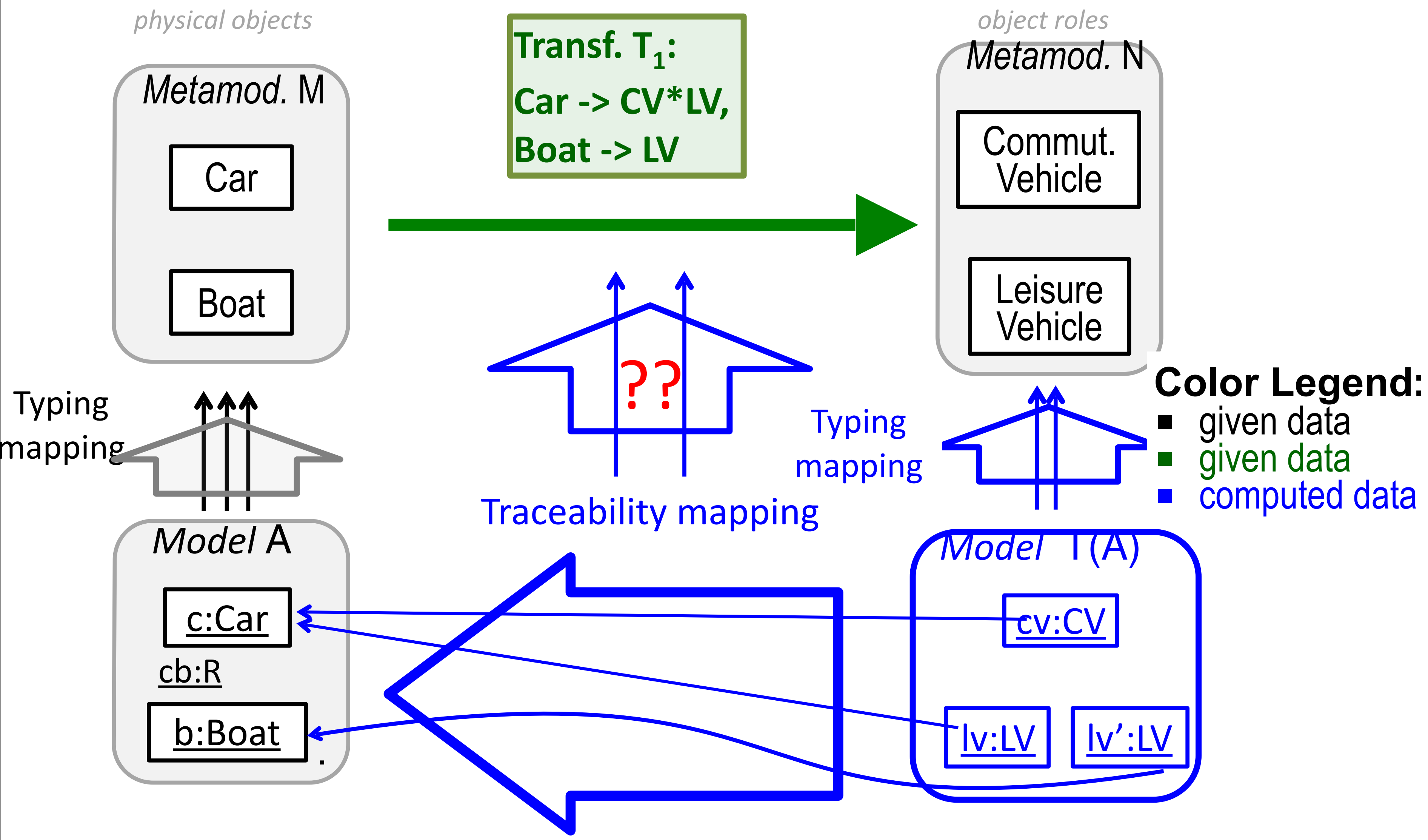
are two-valued (instance x map) functions generated by $T_{1,2}$

Summary 1: Mappings

- Traceability mappings are a **semantic** rather than just **technological** component of MTs
- Provide several **benefits**:
  - hold **useful info** about MTs
    - carry basic Boolean operations
  - help to **understand** MTs
- Should be treated as first-class citizens



Typing: What we have

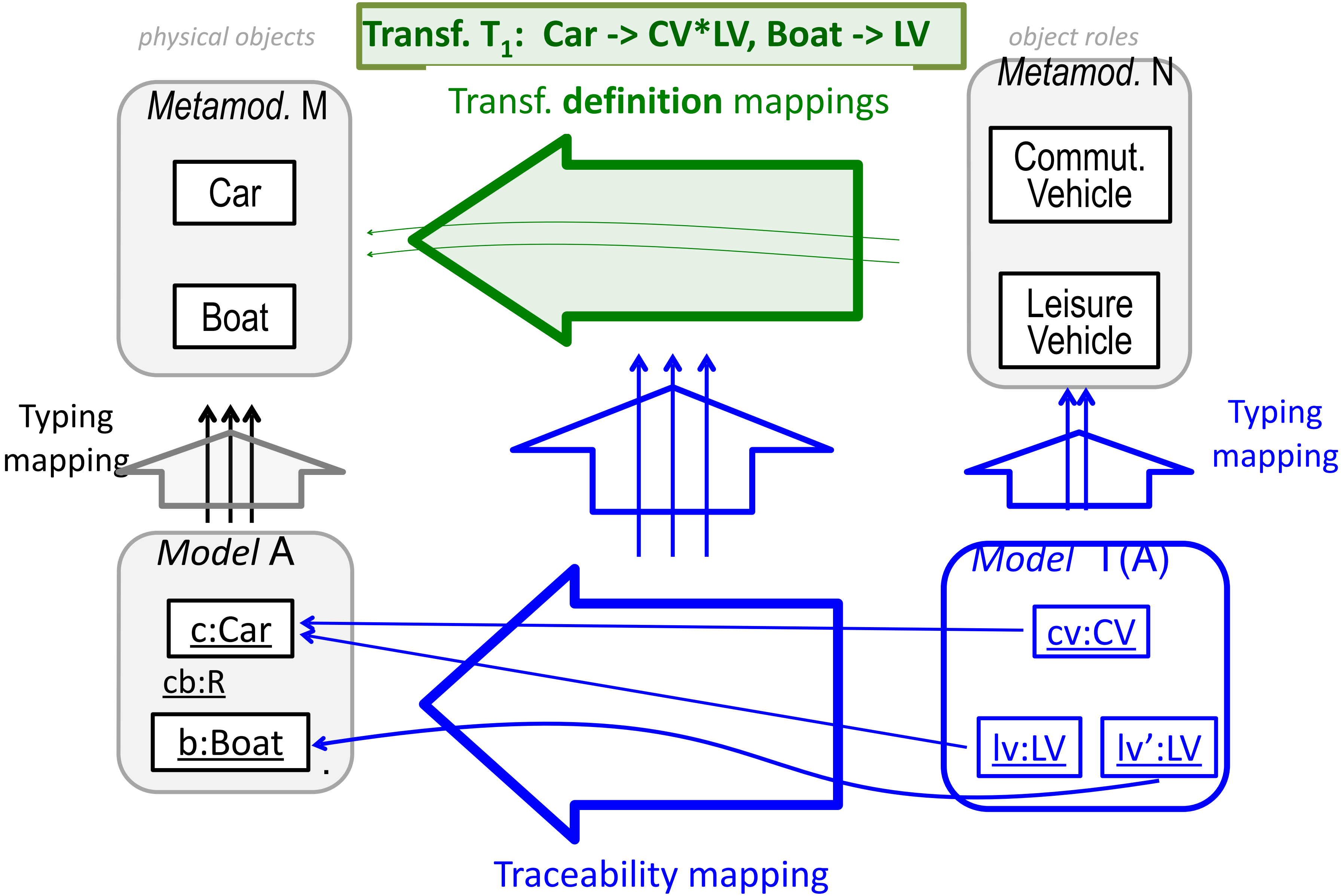


Transf. T_1:
Car -> CV*LV,
Boat -> LV

Color Legend:

- ■ given data
- 🟩 given data
- 🟦 computed data

Typing: What we want



Transf. T_1: Car -> CV*LV, Boat -> LV

physical objects

object roles

Metamod. M

Car

Boat

Transf. **definition** mappings

Metamod. N

Commut. Vehicle

Leisure Vehicle

Typing mapping

Typing mapping

Model A

c:Car

cb:R

b:Boat

.

Model T(A)

cv:CV

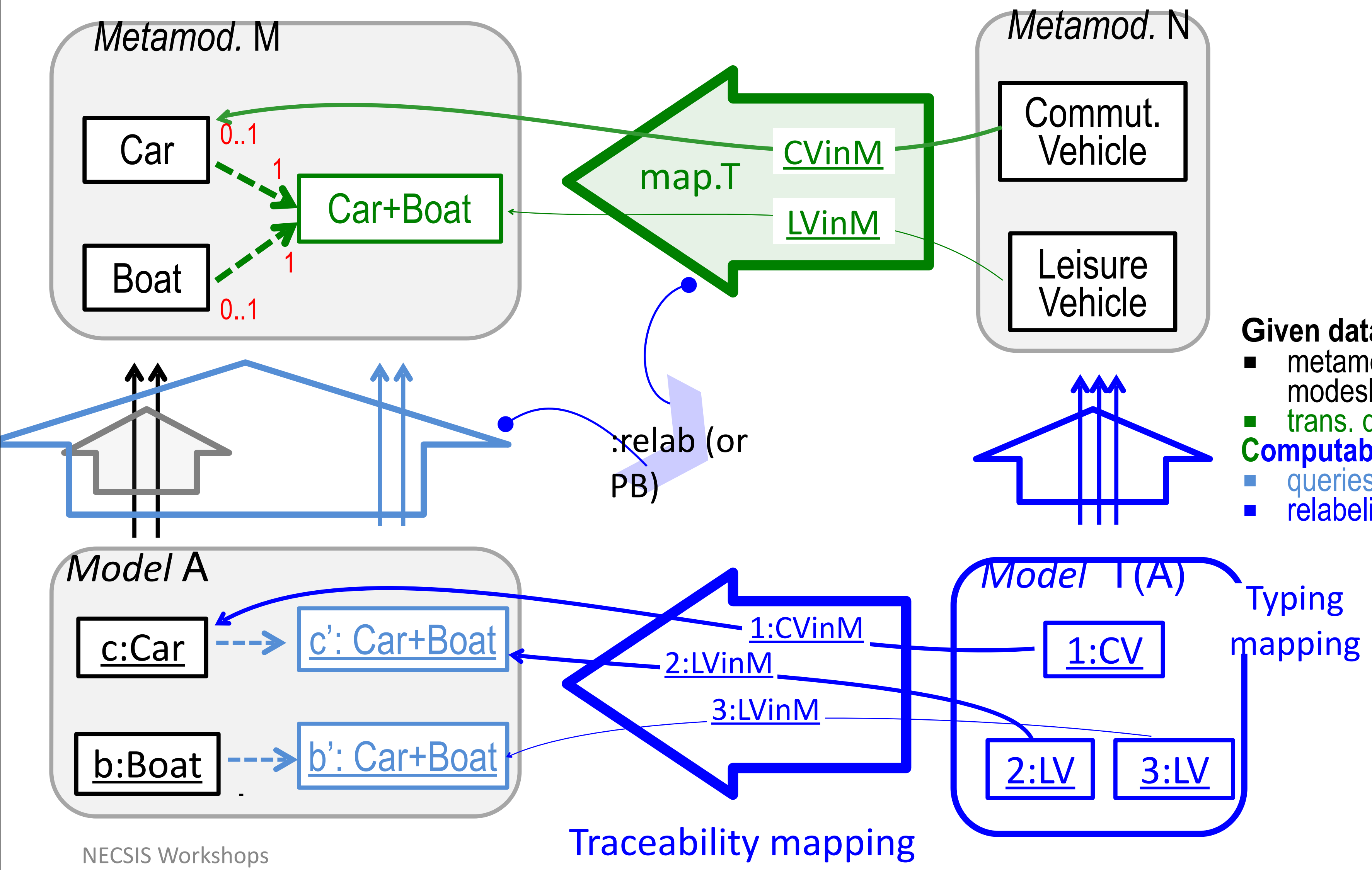
lv:LV

lv':LV

Traceability mapping

Dynamics via mappings: Queries

Transf. T: Car -> CV*LV, Boat -> LV



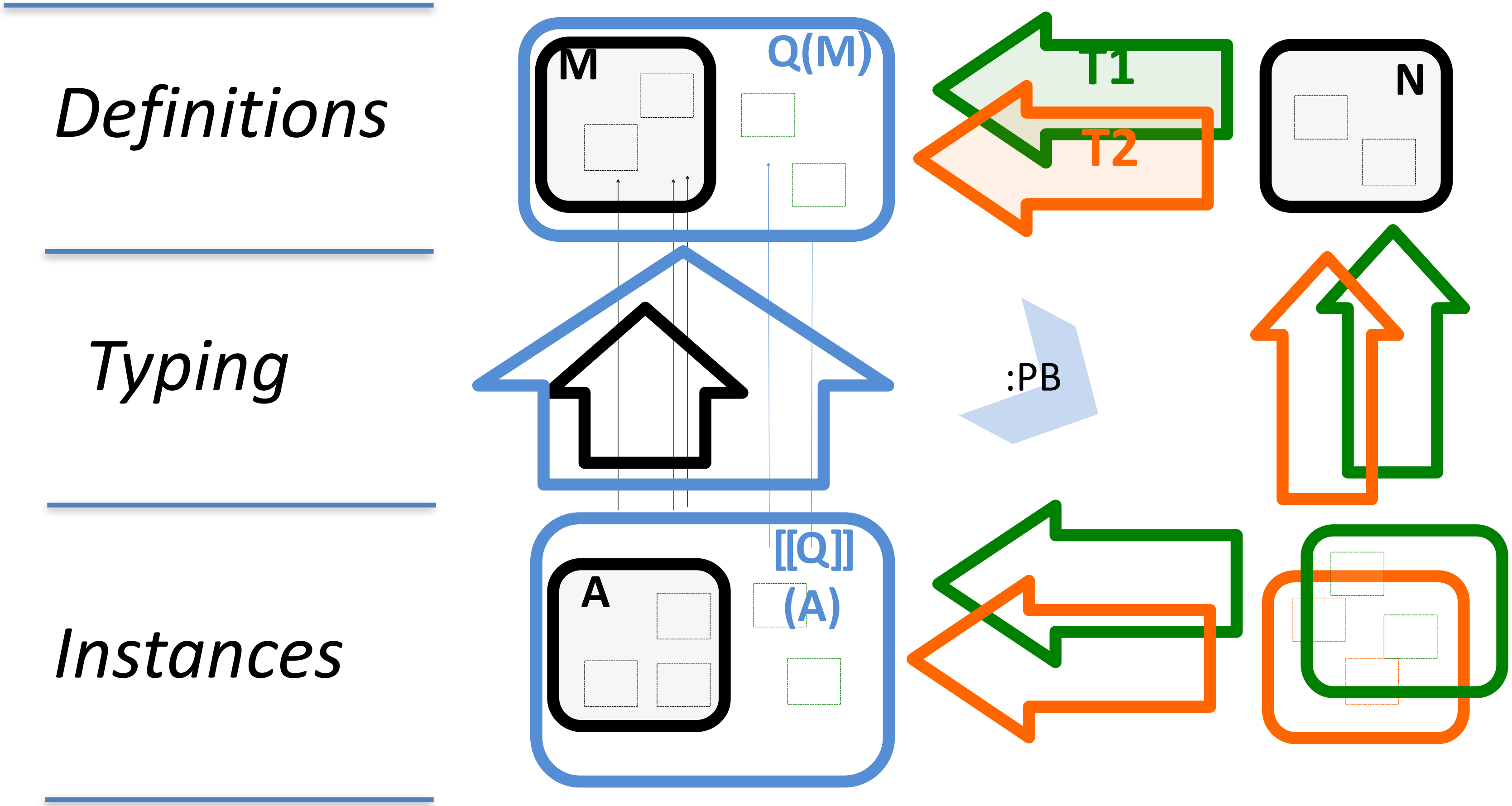
Given data:

- ■ metamodels & modesl
- ■ trans. definition

Computable data:

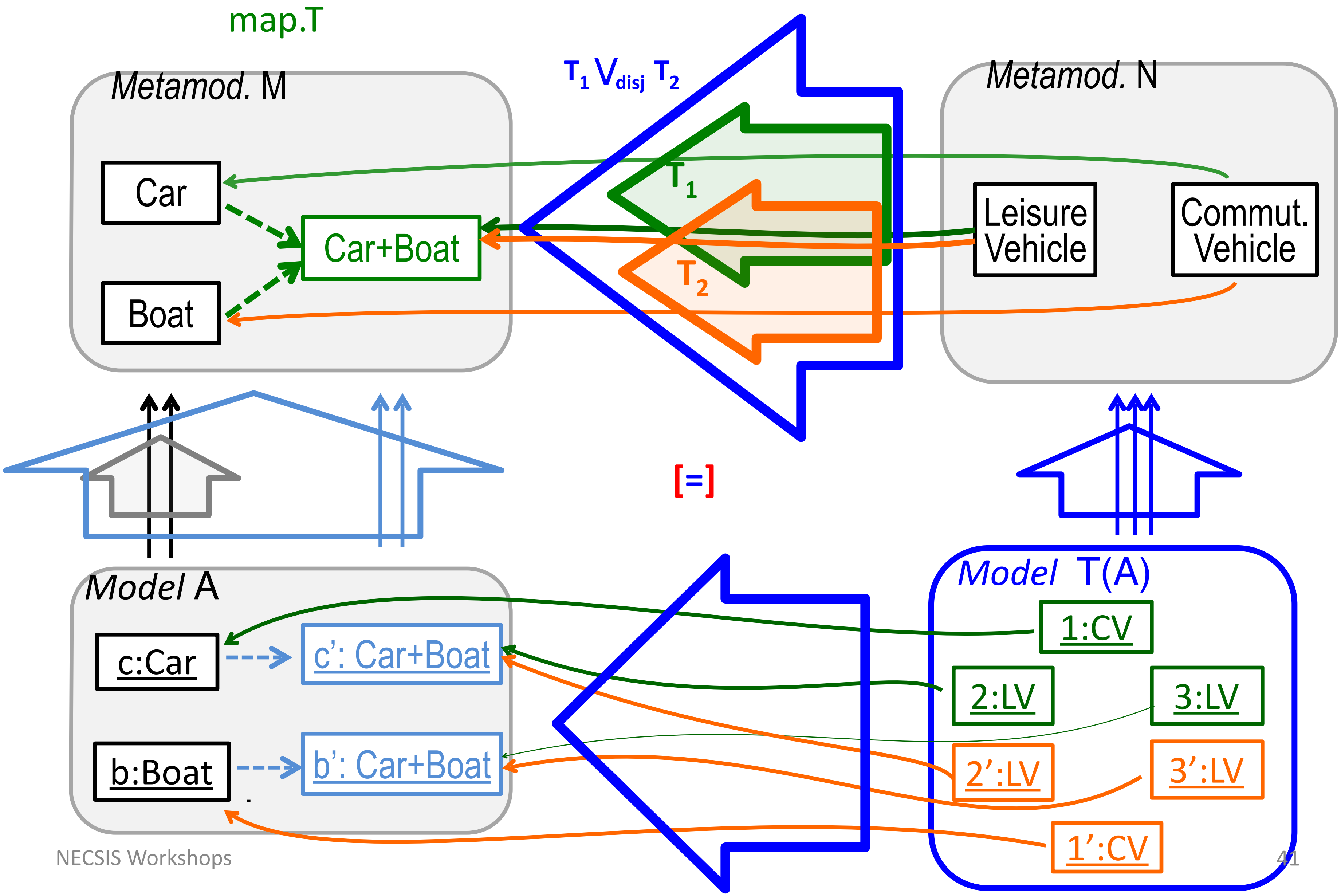
- ■ queries
- ■ relabeling

Algebra and reuse



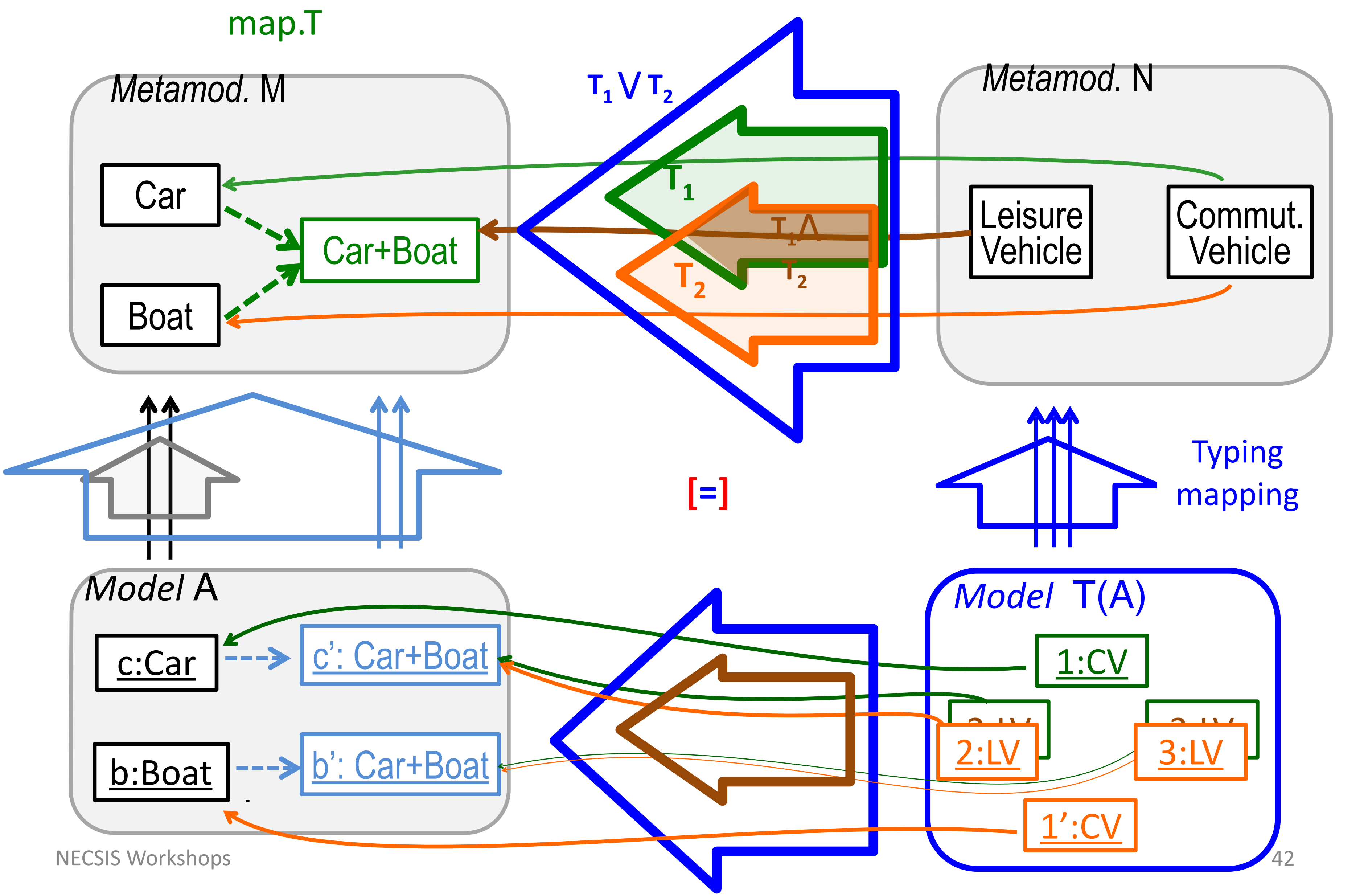
Relabeling as “pulling Q(M) back” **(pullback)**

Algebra of MTs: $T_1 \vee_{disj} T_2$



Algebra of MTs: $T_1 \wedge T_2$ and $T_1 \vee T_2$

> CV*LV, Boat -> LV



map.T

Metamod. M

Car

Car+Boat

Boat

$T_1 \vee T_2$

T_1

$T_1 \wedge T_2$

T_2

Metamod. N

Leisure Vehicle

Commut. Vehicle

[=]

Typing mapping

Model A

c:Car

c': Car+Boat

b:Boat

b': Car+Boat

Model T(A)

1:CV

2:LV

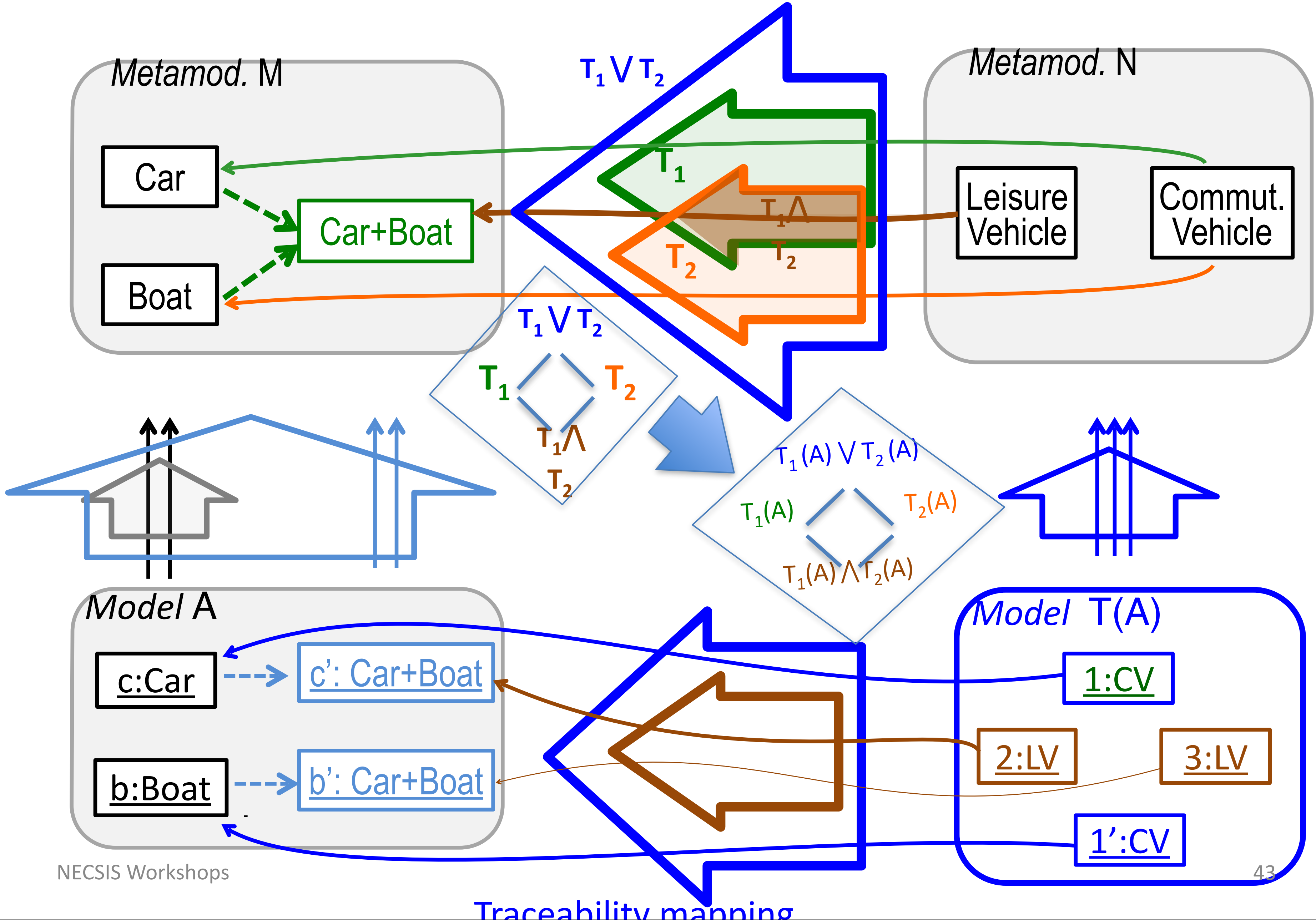
3:LV

2:LV

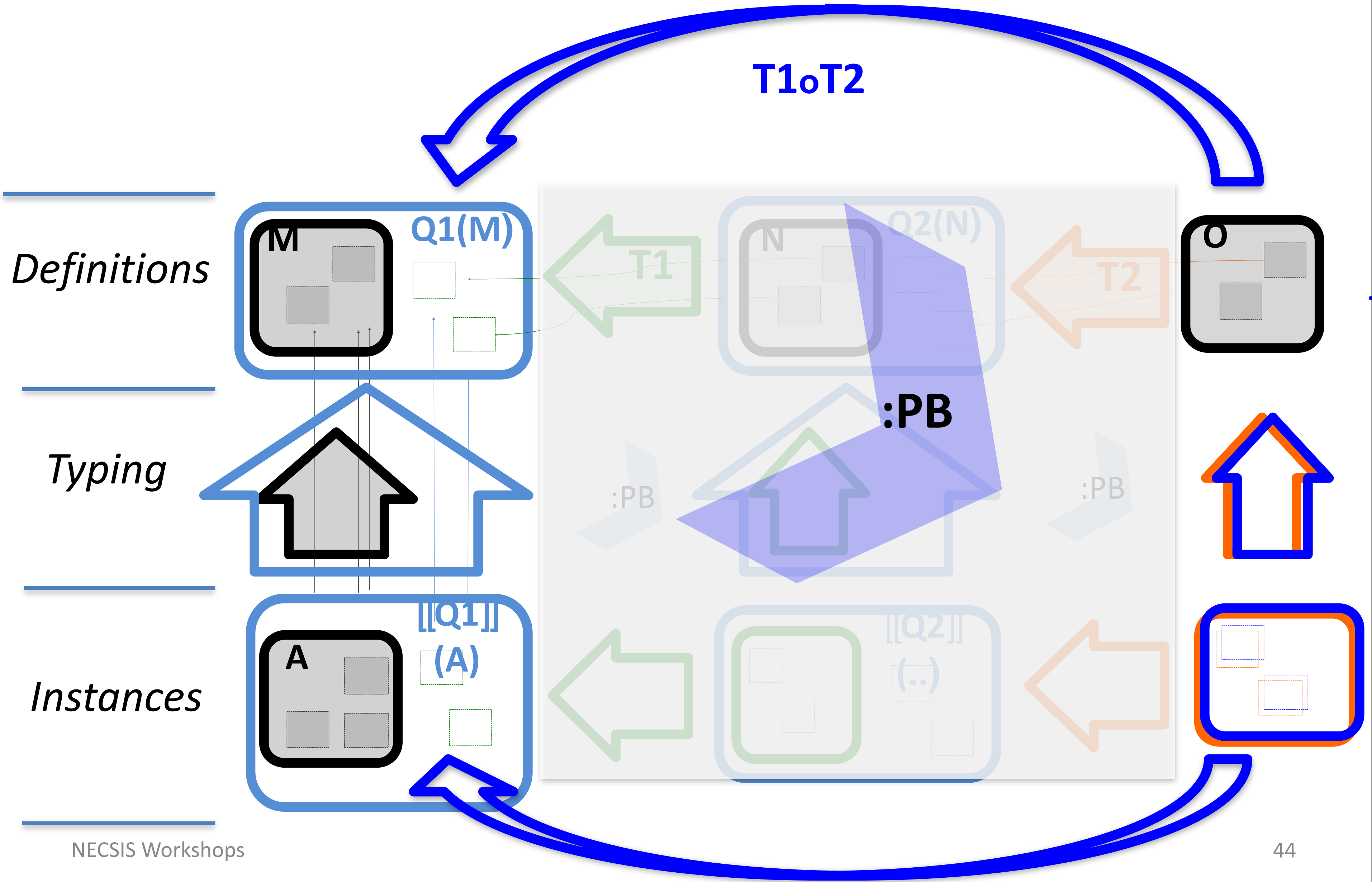
3:LV

1':CV

Algebra of MTs: $T_1 \wedge T_2$ and $T_1 \vee T_2$



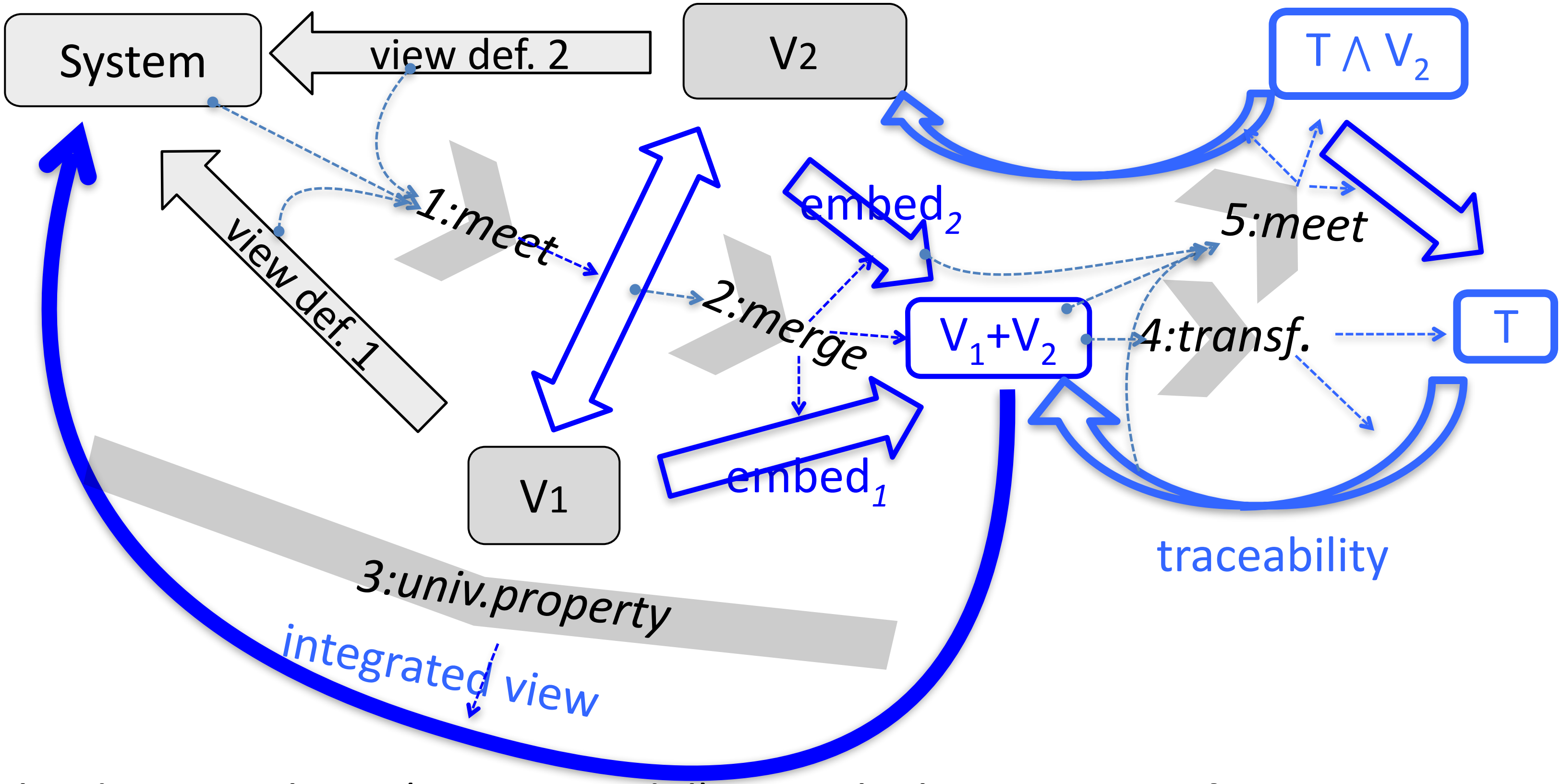
Algebra of MTs. Chaining (seq. composition)



Content

- Intro
- Model merge (BM: choice) via colimit
- Model join (BM: concurrency) via limit
- Model translation via Cartesian monads :)
- Composing operations into workflows

Composing operations into workflows



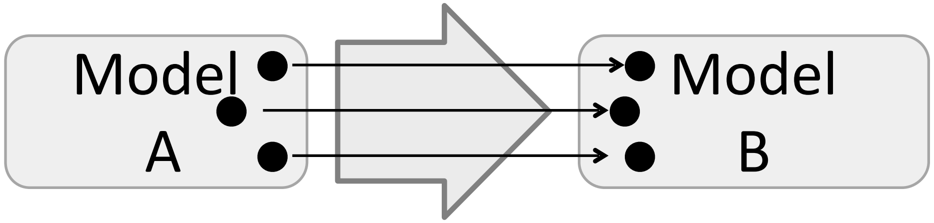
- The diagram above (a megamodel) is an algebraic term in **diagram algebra**

  -- continuity is to be respected!
- Can be executed
- Allow term rewriting (based on laws), hence, optimization

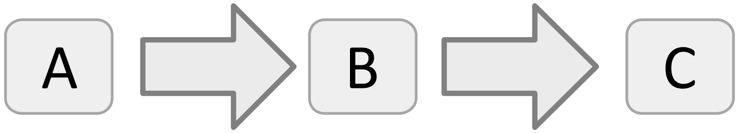
Content

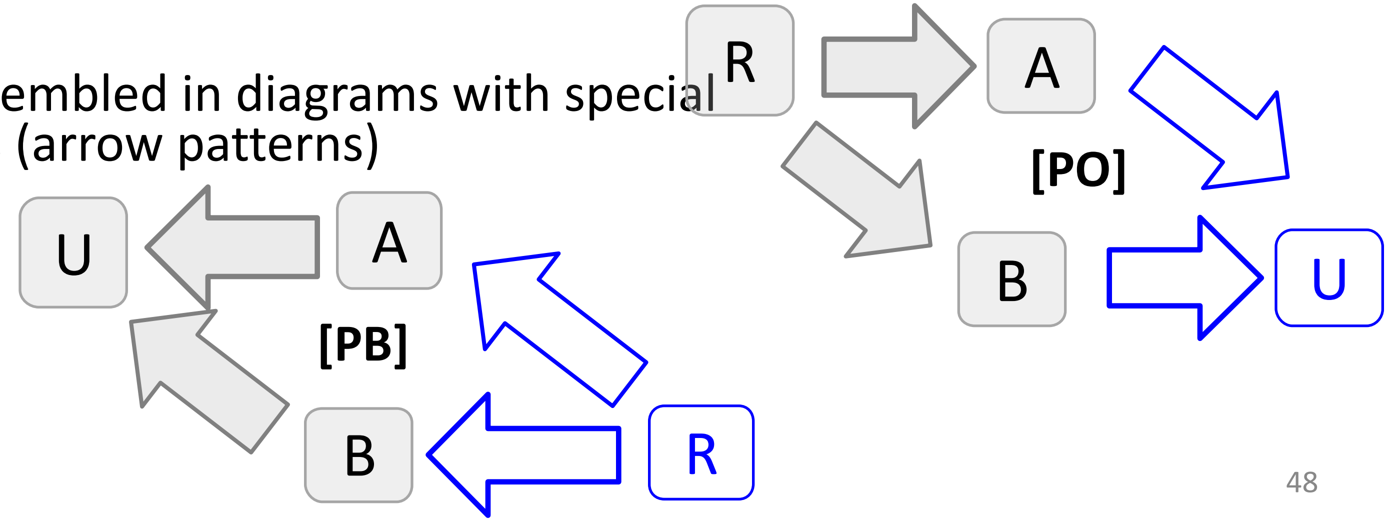
- Intro
- Model merge (BM: choice) via colimit
- Model join (BM: concurrency) via limit
- Model translation via Cartesian monads :)
- Composing operations into workflows
- Summary

Two Dimensions of Mappings



- Mappings are **sets** of links
- Mappings are **directed** entities
  - composable
  - can be assembled in diagrams with special properties (arrow patterns)





Mapping Management

- Model Management ≈ Mapping Management
- Mapping Management needs
  - – conceptual framework

    MISSING
  - – terminological framework

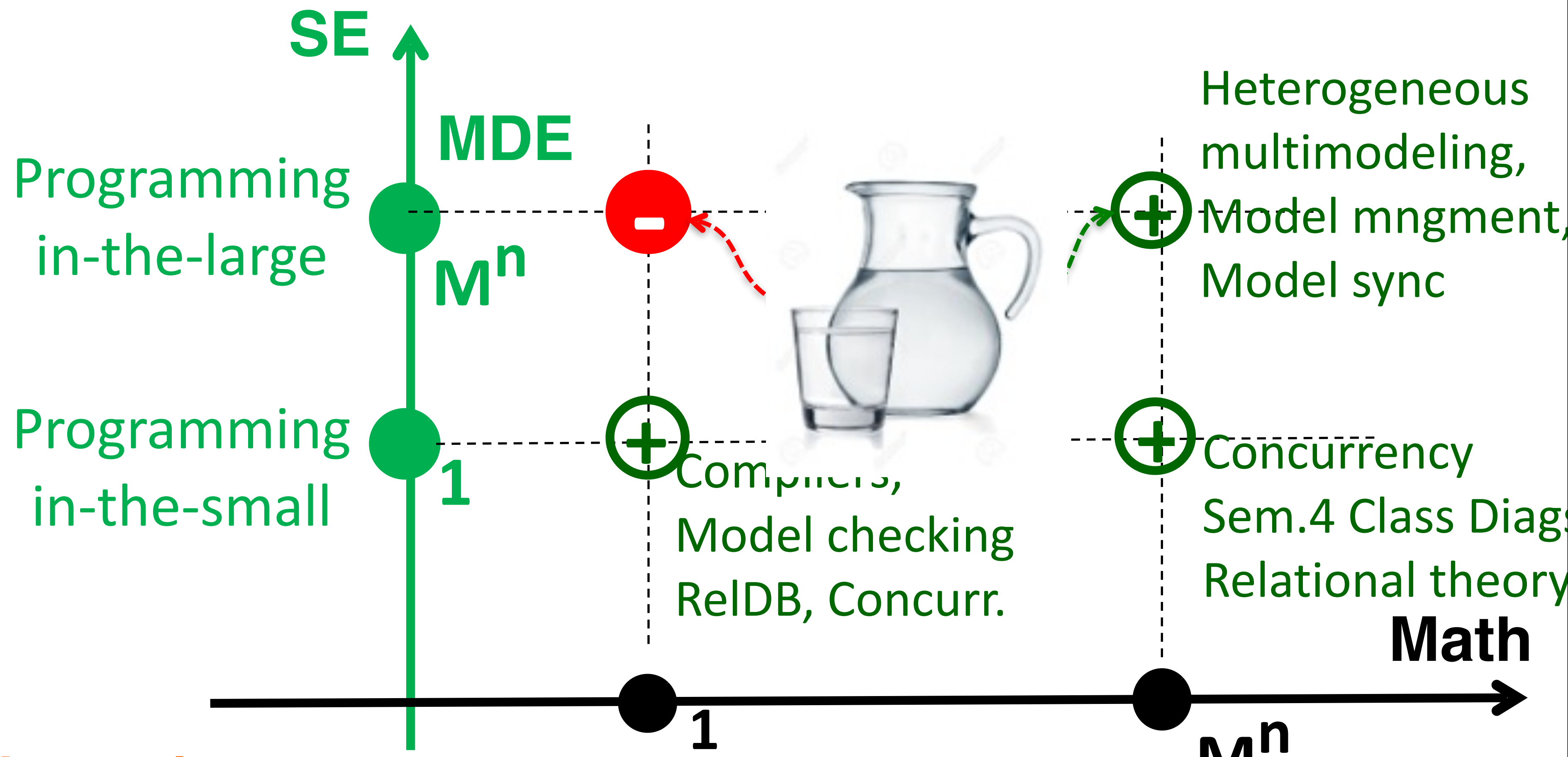
    MISSING
  - – reasonable notation

    MISSING
  - – reasoning techniques

    MISSING
  - – culture of building and manipulating mappings

    MISSING
- Hence the current tooling
- Mathematics of mappings = Category theory

Math for the modern SE



SE

MDE

Programming in-the-large

M^n

Heterogeneous multimodeling, Model mngment, Model sync

Programming in-the-small

1

Compilers, Model checking RelDB, Concurr.

Concurrency Sem.4 Class Diag Relational theory

Math

1

M^n

Classical math

Automata/Grammars Modal logic, FOL

Category theory

Fibrations, Monads, Limits/Colimits, Hyperdoctrines

Legend:

1 – building **one** structure/ module is an issue

M – many: **composition** is the main issue

M^n – many of many

Content

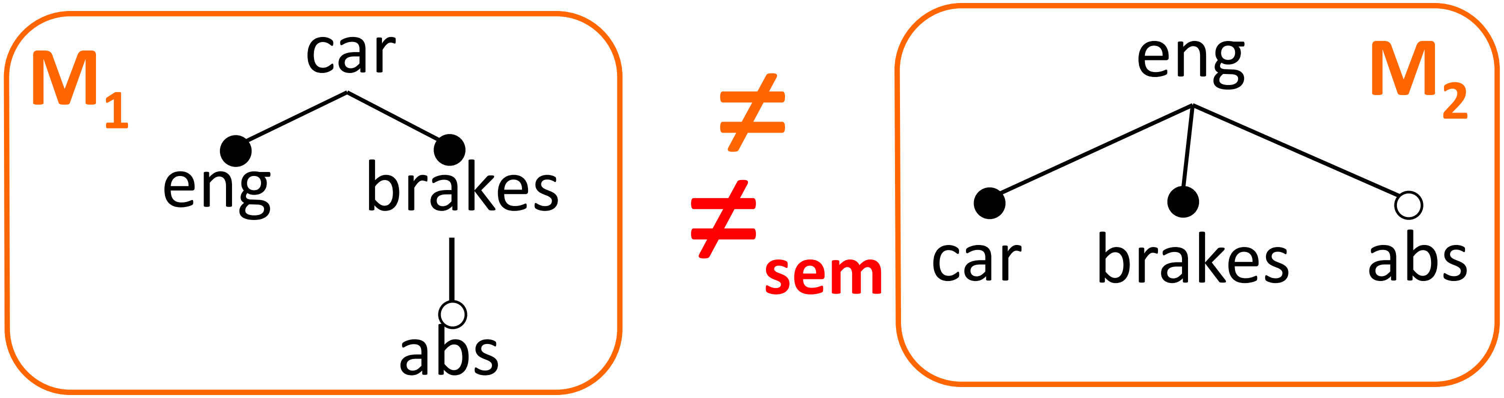
- Specification patterns for model management (18-20 min)
  - Model merge (Beh. modeling: choice) (15 min)
  - Model match (Beh. modeling: concurrency) (5 min)
  - Relational algebra for source-to-target MT (22 min)
- Incremental BX and their taxonomy (0 min b/c of the upcoming NECSIS webinar on Mar 20)
- Foundations of feature modeling (8-10 min)*

*) Does not use category theory :)

Modeling Product Lines with Kripke Structures, Modal Logic and Formal Languages

Ali Safilian[1],
Shoham Ben-David[2,3],
Tom Maibaum[1],
Zinovy Diskin[1,2]
[1] McMaster [2] Waterloo [3] GM

What's (if anything) wrong with Boolean semantics



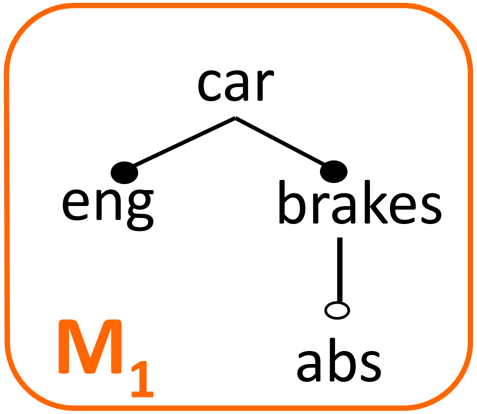
Boolean semantics

$P1 = \{car, eng, brakes\}$
$P2 = P1 \cup \{abs\}$
$\mathbf{PL(M_1)} = \{P1, P2\}$

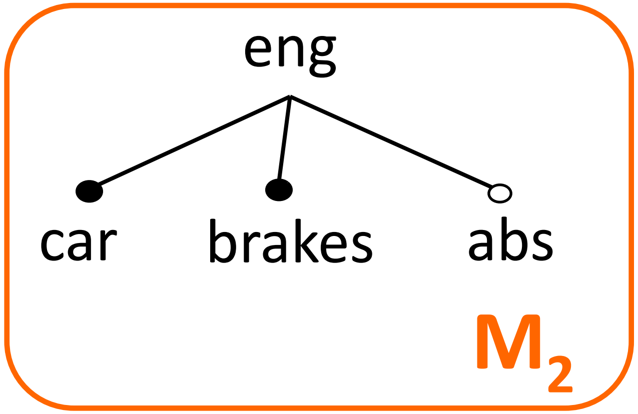
=

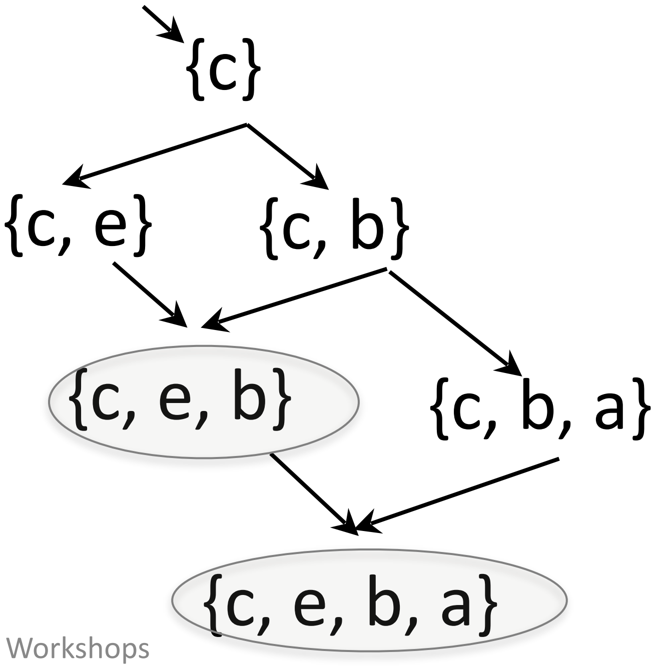
$P1 = \{eng, car, brakes\}$
$P2 = P1 \cup \{abs\}$
$\mathbf{PL(M_2)} = \{P1, P2\}$

FMs and their PPL semantics

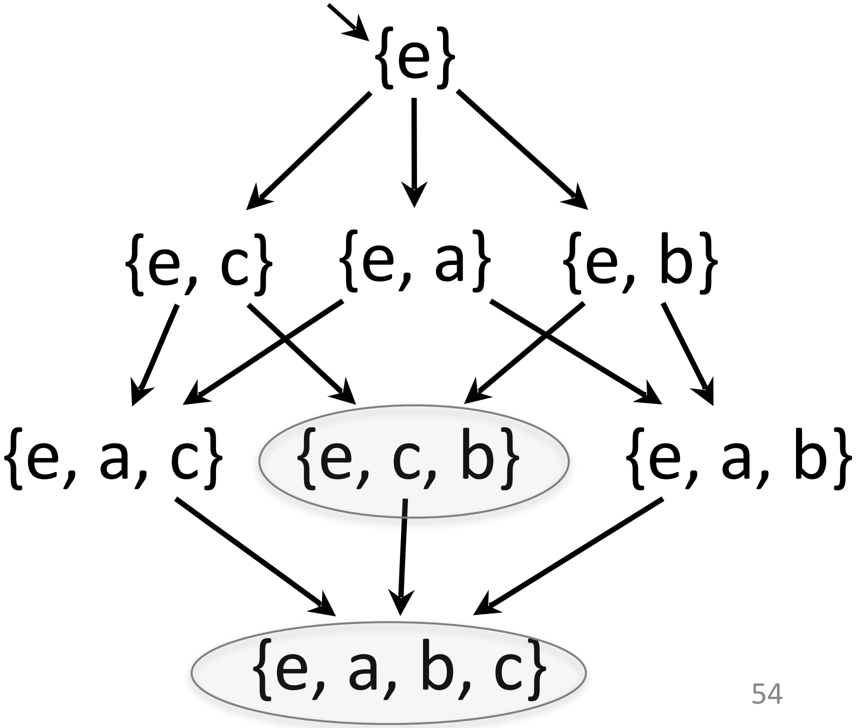

car
eng
brakes
abs
M_1

$\neq_{sem}$

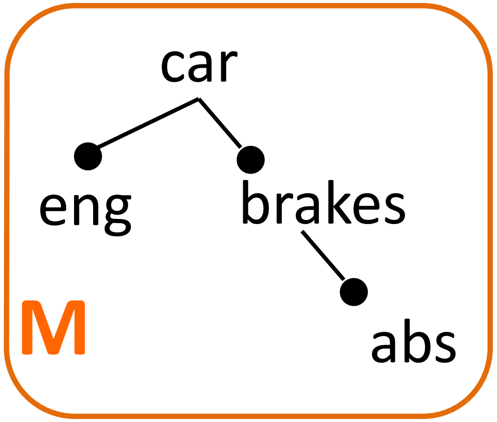

eng
car
brakes
abs
M_2

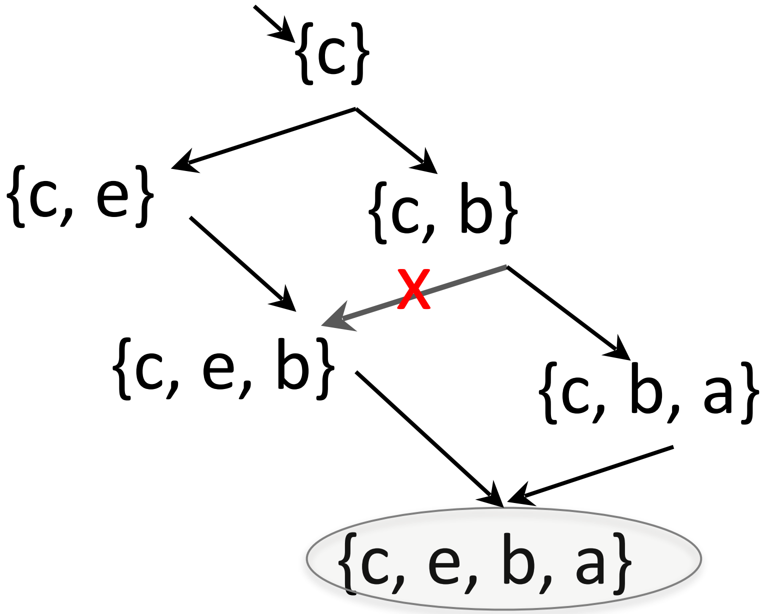

{c}
{c, e}
{c, b}
{c, e, b}
{c, b, a}
{c, e, b, a}

$\neq$

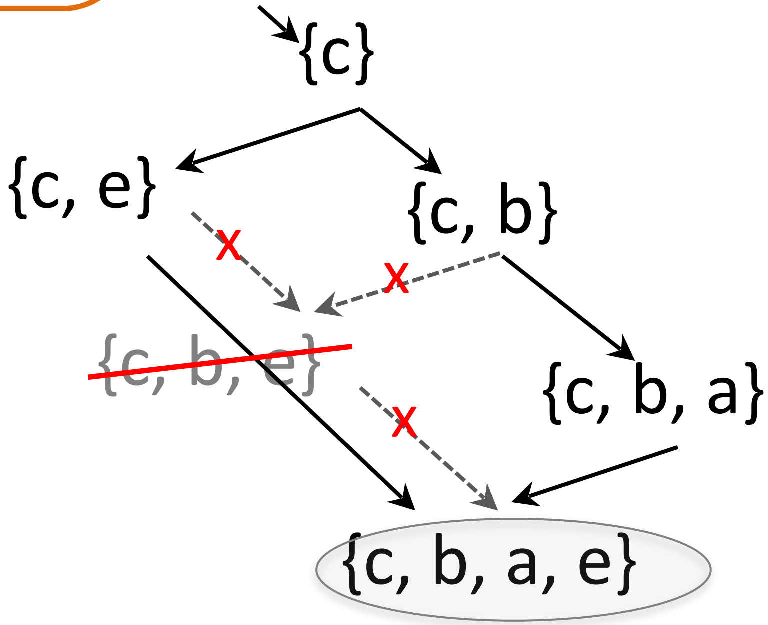

{e}
{e, c}
{e, a}
{e, b}
{e, a, c}
{e, c, b}
{e, a, b}
{e, a, b, c}

Instantiate to completion (I2C)



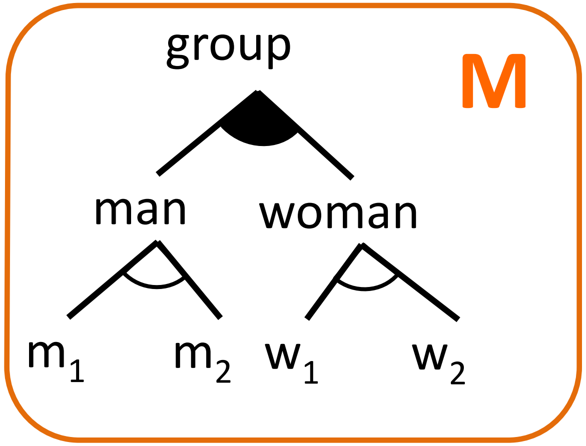


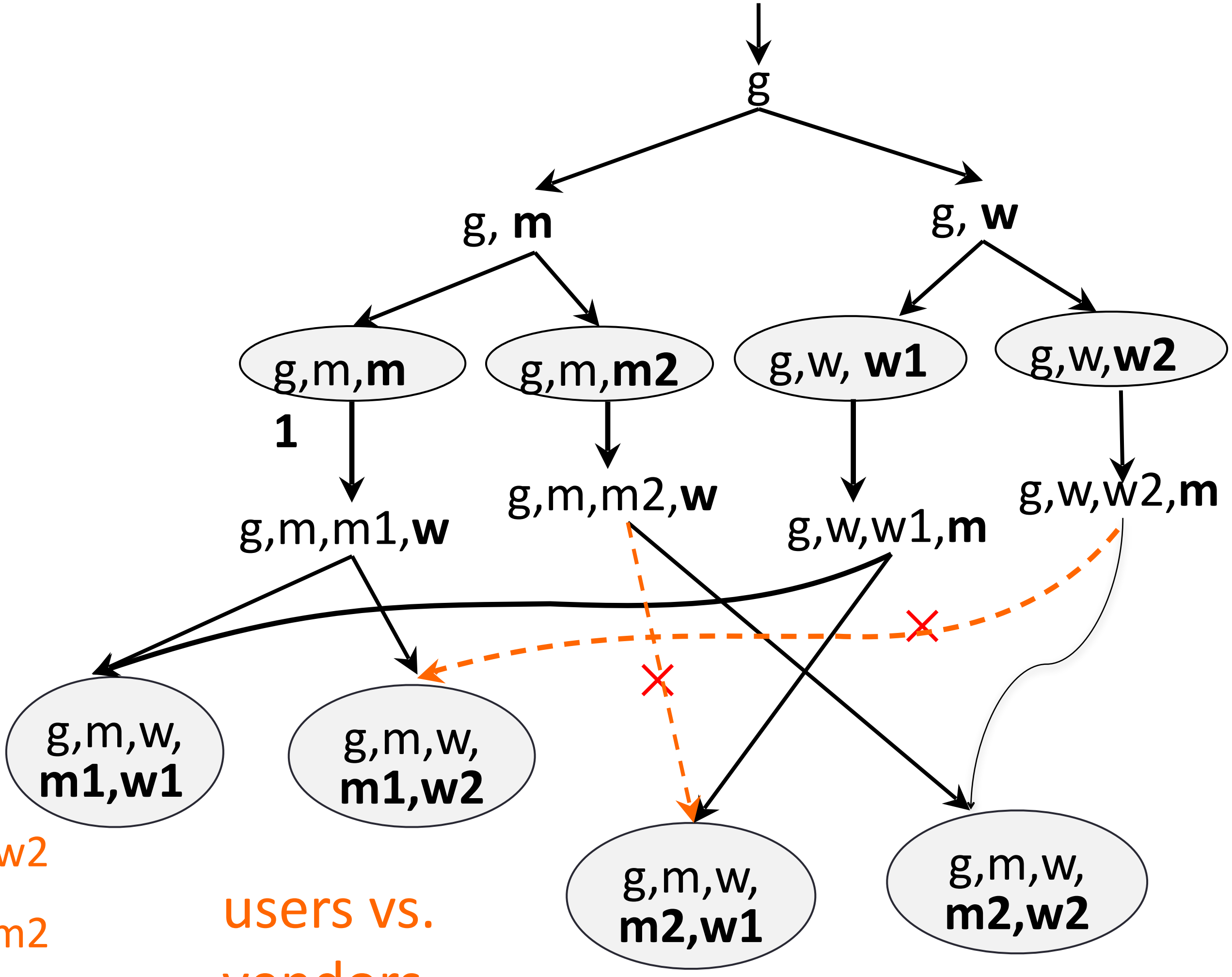
Weak I2C



Strong I2C

Logic and semantics: fCTL and fKS





K |= **!** --> w ∨ m

K |= w --> **AX** (w1 ∨ w2)

K |= (m2 ∧ w ∧ ~w1) --> **AX** w2

K |= (w2 ∧ m ∧ ~m1) --> **AX** m2

users vs.
vendors

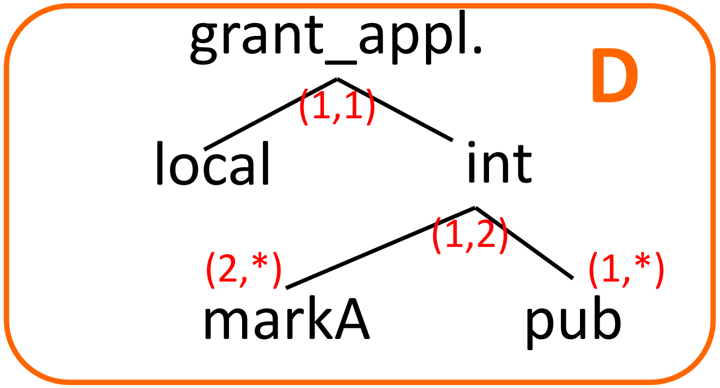
Results

- Any fm M (with all CCConstraints) can be translated into an fCTL theory Φ(M)
- Th.1 (soundness): PL(M) |= Φ(M)
- Th.2 (completeness): K |= Φ(M) iff K = PL(M)
  - analysis of FMs => analysis of Φ(M)s
  - => model checking
- Feature modeling ≈ Event-based behavior modeling (in progress)

FM and Formal Languages

FM with cardinalities, cFM

M = (D, C)



R_{appl} = local $+R_{int}$, where

R_{int} = int.$(R_A + R_p + R_A.R_p + R_p.R_A)$

R_A = markA^2.markA*

R_{pub} = pub.pub*

R(D)

Regular Expressions

- A product is a set of strings, a PL is the union of products (a language)
- Algorithm D ---> R(D)
  - PL(D) = Lang R(D)
  - Preserves the hierarchy in D
- C ---> Lang(C)
  - Lang(M) = Lang(R(D)) ∩ Lang(C)
  - A hierarchy of cFM classes (Chomsky)
- FM analysis => FL analysis
  - Off-the-shelf tools
  - Some analysis operations are **not** decidable in all classes of cFMs

Questions?