Towards a Formal Approach to Metamodel Evolution

Florian Mantz[1], Alessandro Rossini[2], Adrian Rutle[1], Yngve Lamo[1], Uwe Wolter[2]

[1]Faculty of Engineering, Bergen University College, Norway

[2]Department of Informatics, University of Bergen, Norway

11 November 2010
Nordic Workshop on Programming Theory, Turku, Finland

Outline

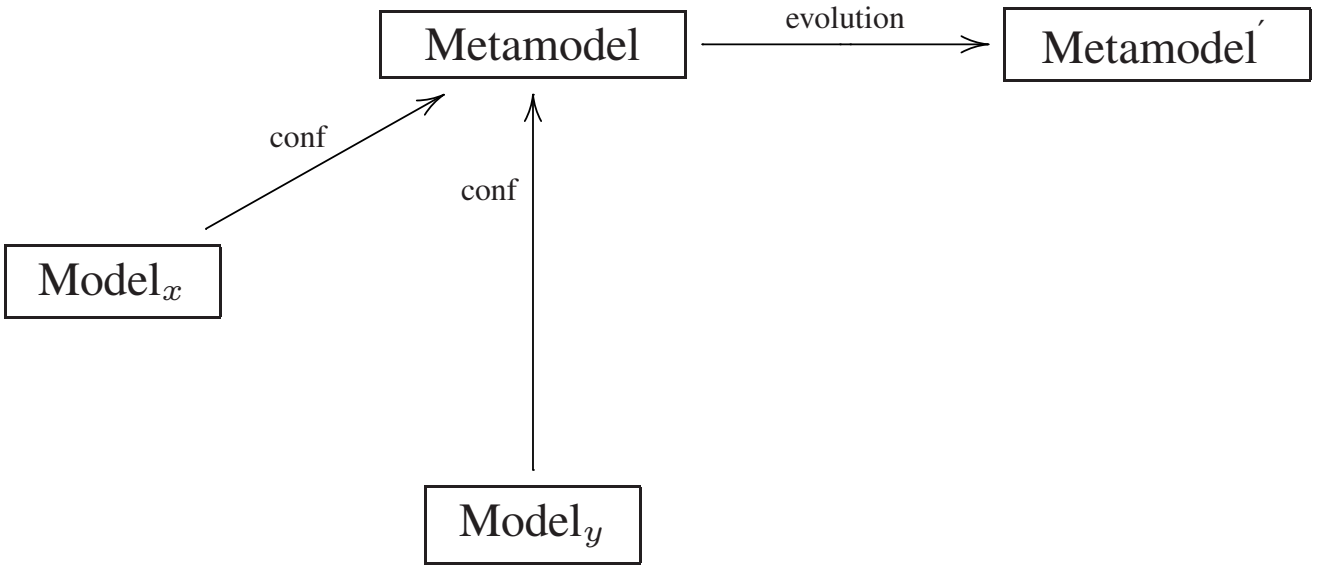
Model-Driven Engineering (MDE)

- Model-driven engineering
  - Models as primary artefacts of software development
  - Model transformations
  - Metamodels as language descriptions
- Complex evolution of models and metamodels
  - Need for techniques and tools to support evolution

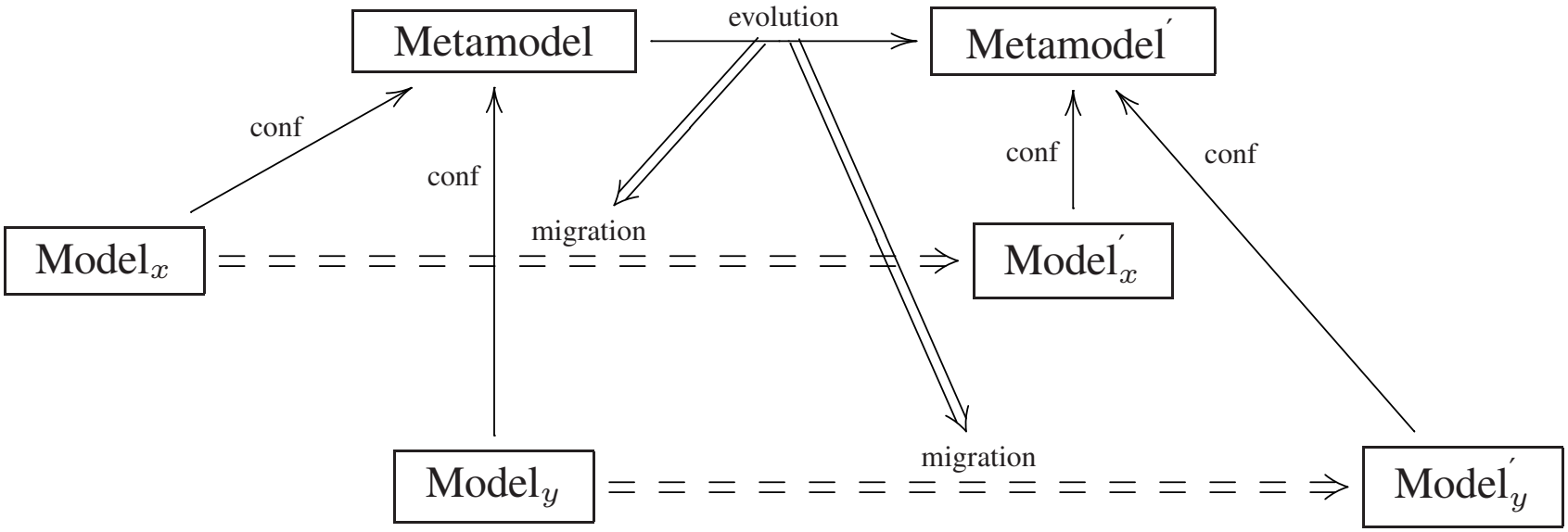
Metamodel Evolution

$$\boxed{\text{Metamodel}} \xrightarrow{\text{evolution}} \boxed{\text{Metamodel}'}$$

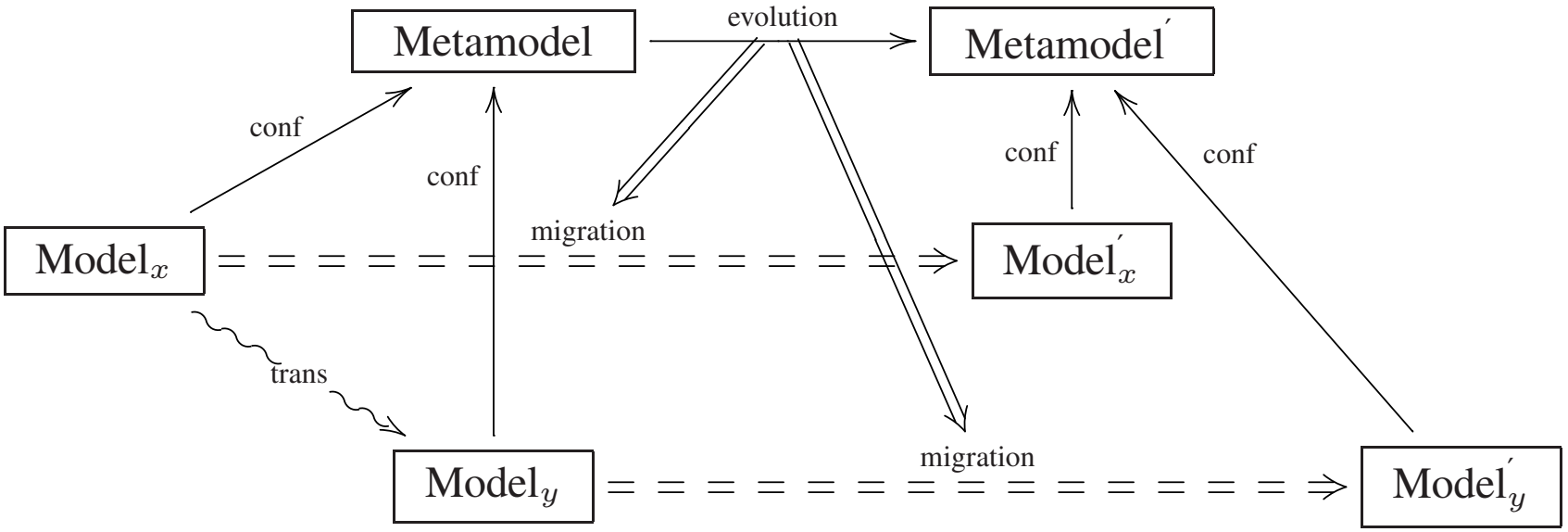
Metamodel Evolution



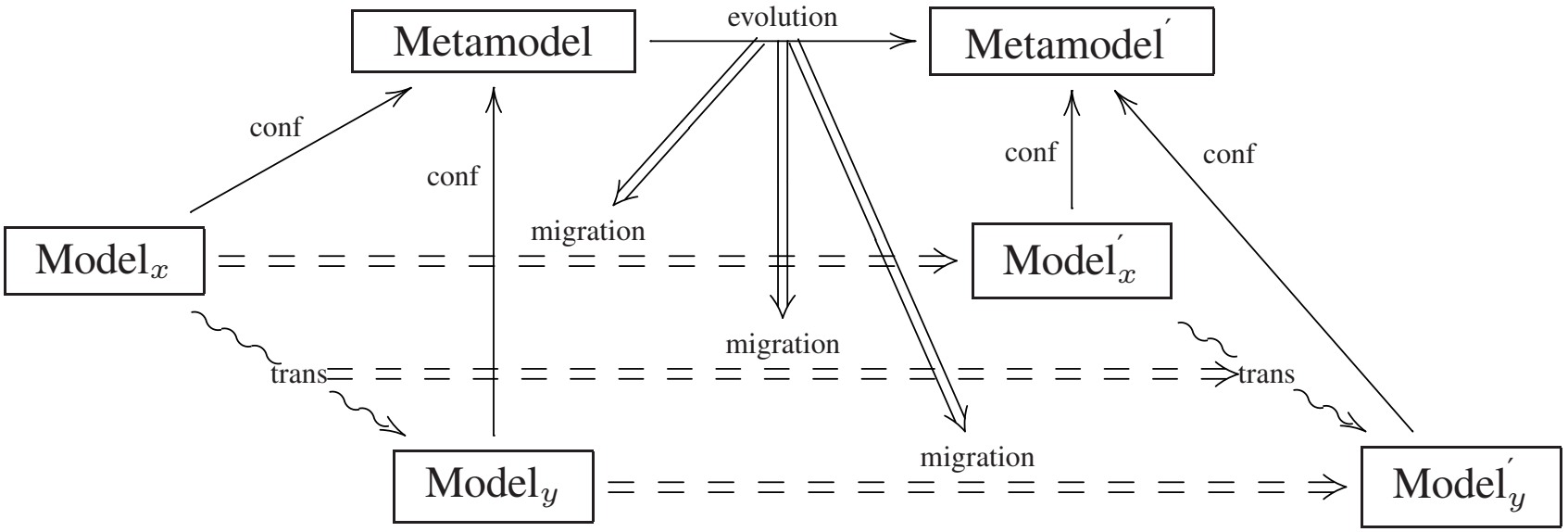
Metamodel Evolution



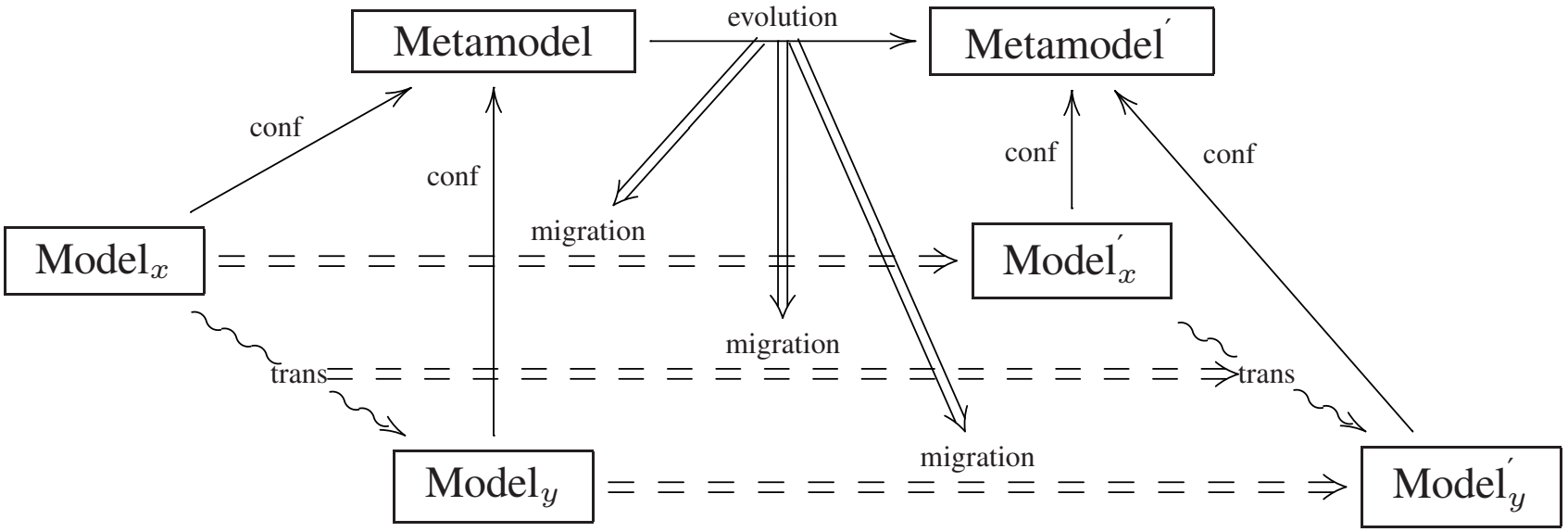
Metamodel Evolution



Metamodel Evolution



Metamodel Evolution



- Metamodels have many dependencies!

Metamodel Evolution Challenges

- Description of metamodel evolution
- Migration of dependent artefacts
  - Models
  - Model transformations
  - Documentations
  - ...
- Lack of formalisation

Diagram Predicate Framework (DPF)

- Formal approach to MDE based on category theory
  - (Meta)modelling
  - Model transformation
  - Model management
- Models in DPF
  - Graph + set of diagrammatic constraints
- Metamodels in DPF
  - Graph + set of diagrammatic constraints

Pushouts & Pullbacks

$$\begin{tikzcd}
 & C \arrow[dl, "s"'] \arrow[dr, "t"] & \\
A \arrow[dr, "s^*"'] & P.O. & B \arrow[dl, "t^*"] \\
 & A \tilde{\cup} B &
\end{tikzcd}$$

Pushout $\approx$ Generalised Union

$$\begin{tikzcd}
 & A \tilde{\cap} B \arrow[dl, "s"'] \arrow[dr, "t"] & \\
A \arrow[dr, "s^*"'] & P.B. & B \arrow[dl, "t^*"] \\
 & C &
\end{tikzcd}$$

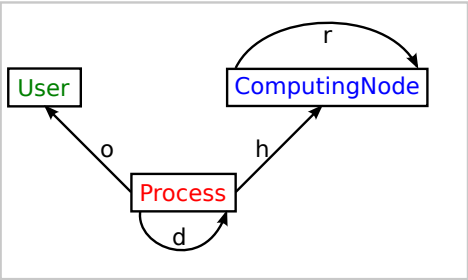
Pullback $\approx$ Generalised Intersection

- Pushouts are used for rule applications
- Pullbacks are used to restrict models

Outline

Sample GRID Metamodel

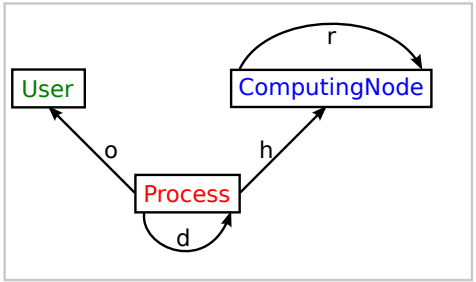
Metamodel



- Modelling language: metamodel + ...

Sample GRID Metamodel

Metamodel

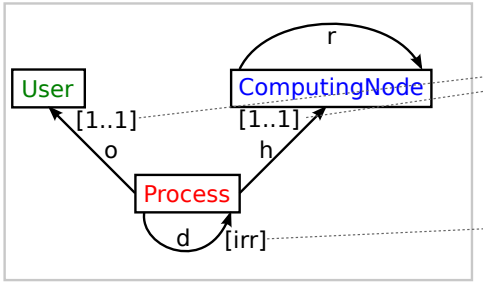


p	$\alpha^{\Sigma_2}(p)$	Proposed vis.	Intended semantics
`[mult(m,n)]`	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[m..n]}]{f}$ Y	$\forall x \in X : m \leq \lvert f(x) \rvert \leq n$, with $0 \leq m \leq n$ and $n \geq 1$
`[surjective]`	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[surj]}]{f}$ Y	$f(X) = Y$
`[irreflexive]`	$1 \circlearrowleft f$	[irr] X $\circlearrowleft$ f	$\forall x \in X : x \quad f(x)$
`[image-inclusion]`	$1 \underset{b}{\overset{a}{\rightrightarrows}} 2$	X $\underset{g}{\overset{f}{\underset{[\sqsubseteq]}{\rightrightarrows}}}$ Y	$\forall x \in X : f(x) \subseteq g(x)$
`[composition]`	$1 \xrightarrow{a} 2$, $2 \xrightarrow{b} 3$, $1 \xrightarrow{c} 3$	X $\xrightarrow{f}$ Y, Y $\xrightarrow{g}$ Z, X $\xrightarrow[\text{[comp]}]{h}$ Z	$\forall x \in X : h(x) = \{g(y) \mid y \in f(x)\}$

- Modelling language: metamodel + signature

Sample GRID Metamodel

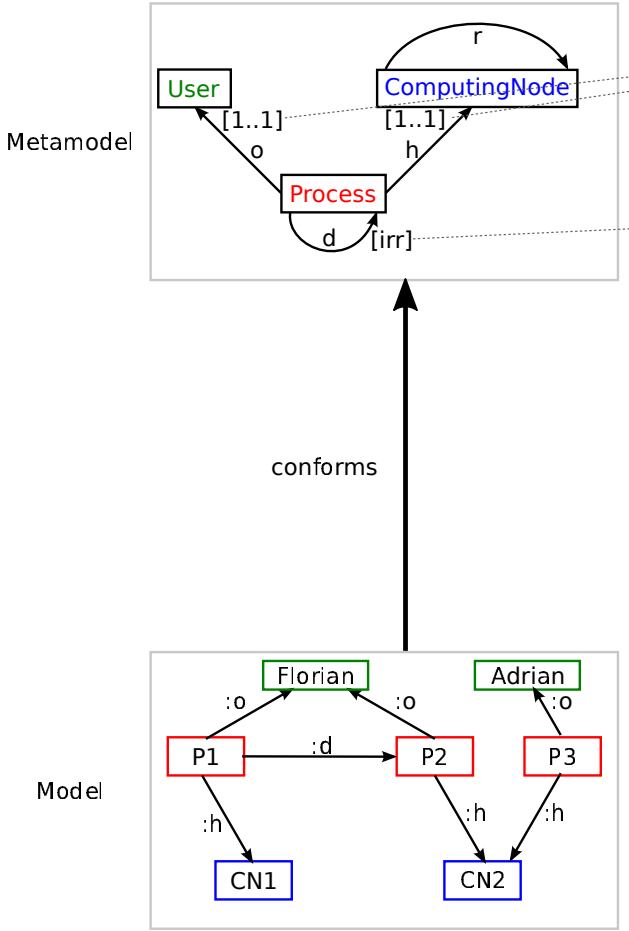
Metamodel



p	$\alpha^{\Sigma_2}(p)$	Proposed vis.	Intended semantics
[mult (m,n)]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[m..n]}]{f}$ Y	$\forall x \in X : m \leq \lvert f(x) \rvert \leq n$, with $0 \leq m \leq n$ and $n \geq 1$
[surjective]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[surj]}]{f}$ Y	$f(X) = Y$
[irreflexive]	$1 \circlearrowleft f$	[irr] X $\circlearrowleft$ f	$\forall x \in X : x \quad f(x)$
[image-inclusion]	$1 \underset{b}{\overset{a}{\rightrightarrows}} 2$	X $\underset{g}{\overset{f}{\underset{[\sqsubseteq]}{\rightrightarrows}}}$ Y	$\forall x \in X : f(x) \subseteq g(x)$
[composition]	$1 \xrightarrow{a} 2$, $2 \xrightarrow{b} 3$, $1 \xrightarrow{c} 3$	X $\xrightarrow{f}$ Y, Y $\xrightarrow{g}$ Z, X $\xrightarrow[\text{[comp]}]{h}$ Z	$\forall x \in X : h(x) = \{g(y) \mid y \in f(x)\}$

- Req 1: “a process is owned exactly by one user”
- Req 2: “a process runs exactly on one computing node”
- Req 3: “a process cannot depend on itself”

Sample GRID Metamodel

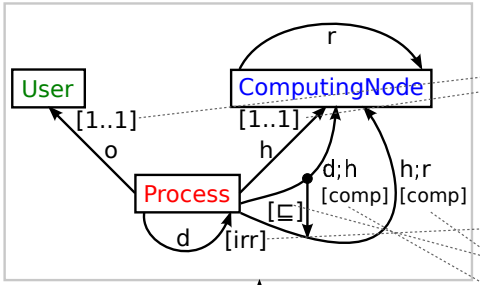


p	$\alpha^{\Sigma_2}(p)$	Proposed vis.	Intended semantics
[mult (m,n)]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[m..n]}]{f}$ Y	$\forall x \in X : m \leq \lvert f(x) \rvert \leq n$, with $0 \leq m \leq n$ and $n \geq 1$
[surjective]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[surj]}]{f}$ Y	$f(X) = Y$
[irreflexive]	$1 \circlearrowleft f$	[irr] X $\circlearrowleft$ f	$\forall x \in X : x \quad f(x)$
[image-inclusion]	$1 \underset{b}{\overset{a}{\rightrightarrows}} 2$	X $\underset{g}{\overset{f}{\underset{[\sqsubseteq]}{\rightrightarrows}}}$ Y	$\forall x \in X : f(x) \subseteq g(x)$
[composition]	$1 \xrightarrow{a} 2$, $2 \xrightarrow{b} 3$, $1 \xrightarrow{c} 3$	X $\xrightarrow{f}$ Y, Y $\xrightarrow{g}$ Z, X $\xrightarrow[\text{[comp]}]{h}$ Z	$\forall x \in X : h(x) = \{g(y) \mid y \in f(x)\}$

- Models have to conform to their metamodel

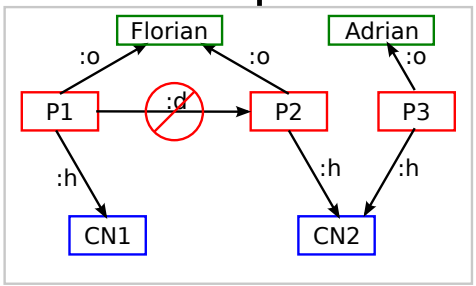
Sample GRID Metamodel

Metamodel



conforms

Model



p	$\alpha^{\Sigma_2}(p)$	Proposed vis.	Intended semantics
[mult (m,n)]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[m..n]}]{f}$ Y	$\forall x \in X : m \leq \lvert f(x) \rvert \leq n$, with $0 \leq m \leq n$ and $n \geq 1$
[surjective]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[surj]}]{f}$ Y	$f(X) = Y$
[irreflexive]	$1 \circlearrowleft f$	[irr] X $\circlearrowleft$ f	$\forall x \in X : x \quad f(x)$
[image-inclusion]	$1 \underset{b}{\overset{a}{\rightrightarrows}} 2$	X $\underset{g}{\overset{f}{\underset{\text{[⊑]}}{\rightrightarrows}}}$ Y	$\forall x \in X : f(x) \subseteq g(x)$
[composition]	$1 \xrightarrow{a} 2$, $2 \xrightarrow{b} 3$, $1 \xrightarrow{c} 3$	X $\xrightarrow{f}$ Y, Y $\xrightarrow{g}$ Z, X $\xrightarrow[\text{[comp]}]{h}$ Z	$\forall x \in X : h(x) = \quad \{g(y) \mid y \in f(x)\}$

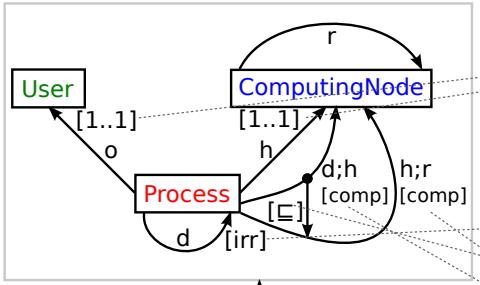
New Constraint:

A Process P running on ComputingNode CN can only depend on a Process P' running on ComputingNode CN' if ComputingNode CN' is reachable by ComputingNode CN.

- The metamodel may be revised, e.g. new constraints may be added
- This may break the conformance of some models

Sample GRID Metamodel

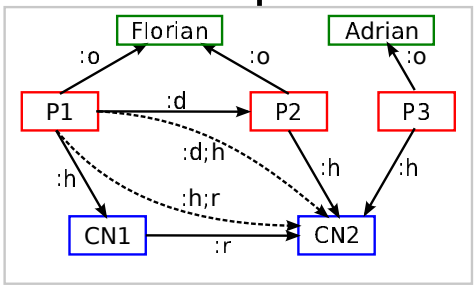
Metamodel



p	$\alpha^{\Sigma_2}(p)$	Proposed vis.	Intended semantics
[mult (m,n)]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[m..n]}]{f}$ Y	$\forall x \in X : m \leq \lvert f(x) \rvert \leq n$, with $0 \leq m \leq n$ and $n \geq 1$
[surjective]	$1 \xrightarrow{a} 2$	X $\xrightarrow[\text{[surj]}]{f}$ Y	$f(X) = Y$
[irreflexive]	$1 \circlearrowleft f$	[irr] X $\circlearrowleft$ f	$\forall x \in X : x \quad f(x)$
[image-inclusion]	$1 \underset{b}{\overset{a}{\rightrightarrows}} 2$	X $\underset{g}{\overset{f}{\underset{[\sqsubseteq]}{\rightrightarrows}}}$ Y	$\forall x \in X : f(x) \subseteq g(x)$
[composition]	$1 \xrightarrow{a} 2$, $2 \xrightarrow{b} 3$, $1 \xrightarrow{c} 3$	X $\xrightarrow{f}$ Y, Y $\xrightarrow{g}$ Z, X $\xrightarrow[\text{[comp]}]{h}$ Z	$\forall x \in X : h(x) = \quad \{g(y) \mid y \in f(x)\}$

conforms

Model



New Constraint:

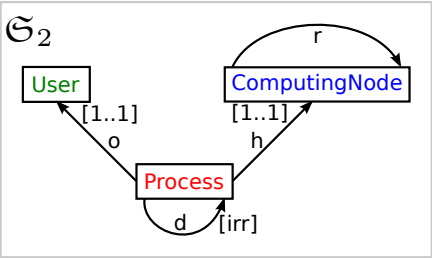
A Process P running on ComputingNode CN can only depend on a Process P' running on ComputingNode CN' if ComputingNode CN' is reachable by ComputingNode CN.

- Models have to be "repaired" in one way or another

Outline

Example

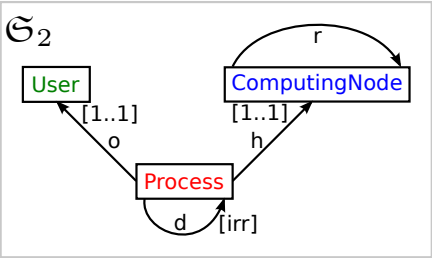
Metamodel



Example

Metamodel Evolution

Add new Constraint



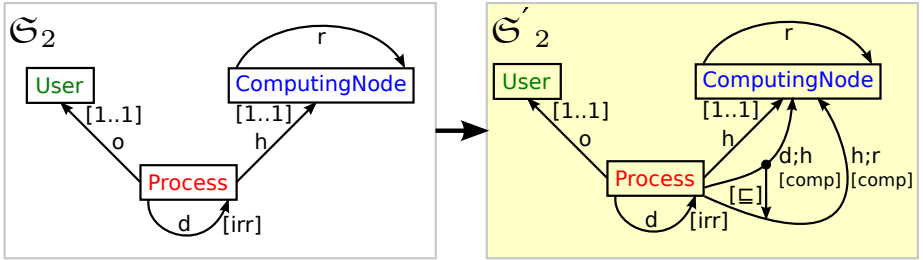
New Constraint:

A Process P running on ComputingNode CN can only depend on a Process P' running on ComputingNode CN' if ComputingNode CN' is reachable by ComputingNode CN.

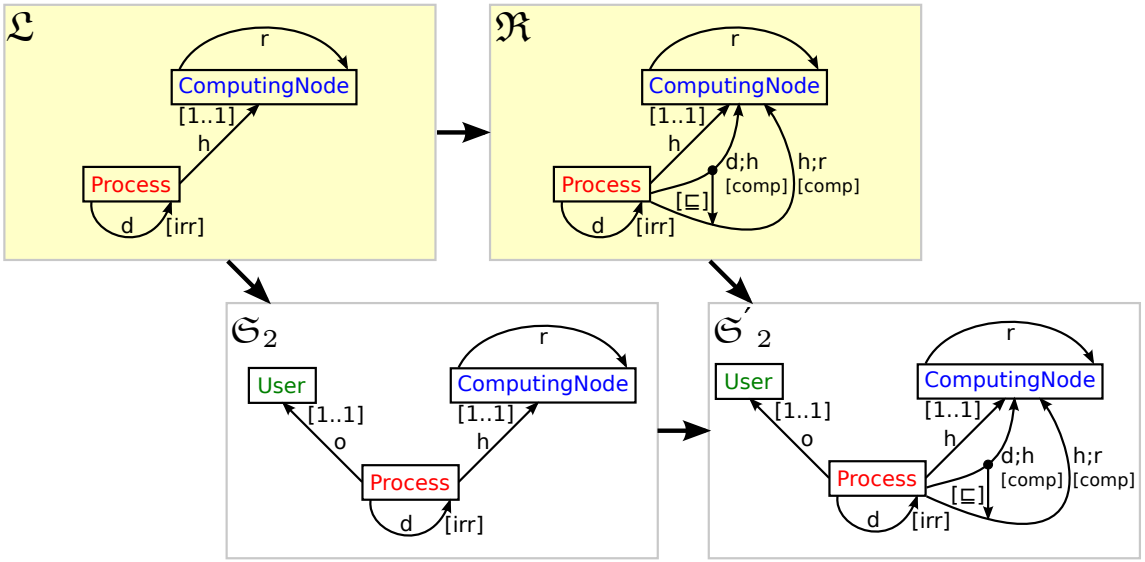
Example

Metamodel Evolution

Add new Constraint



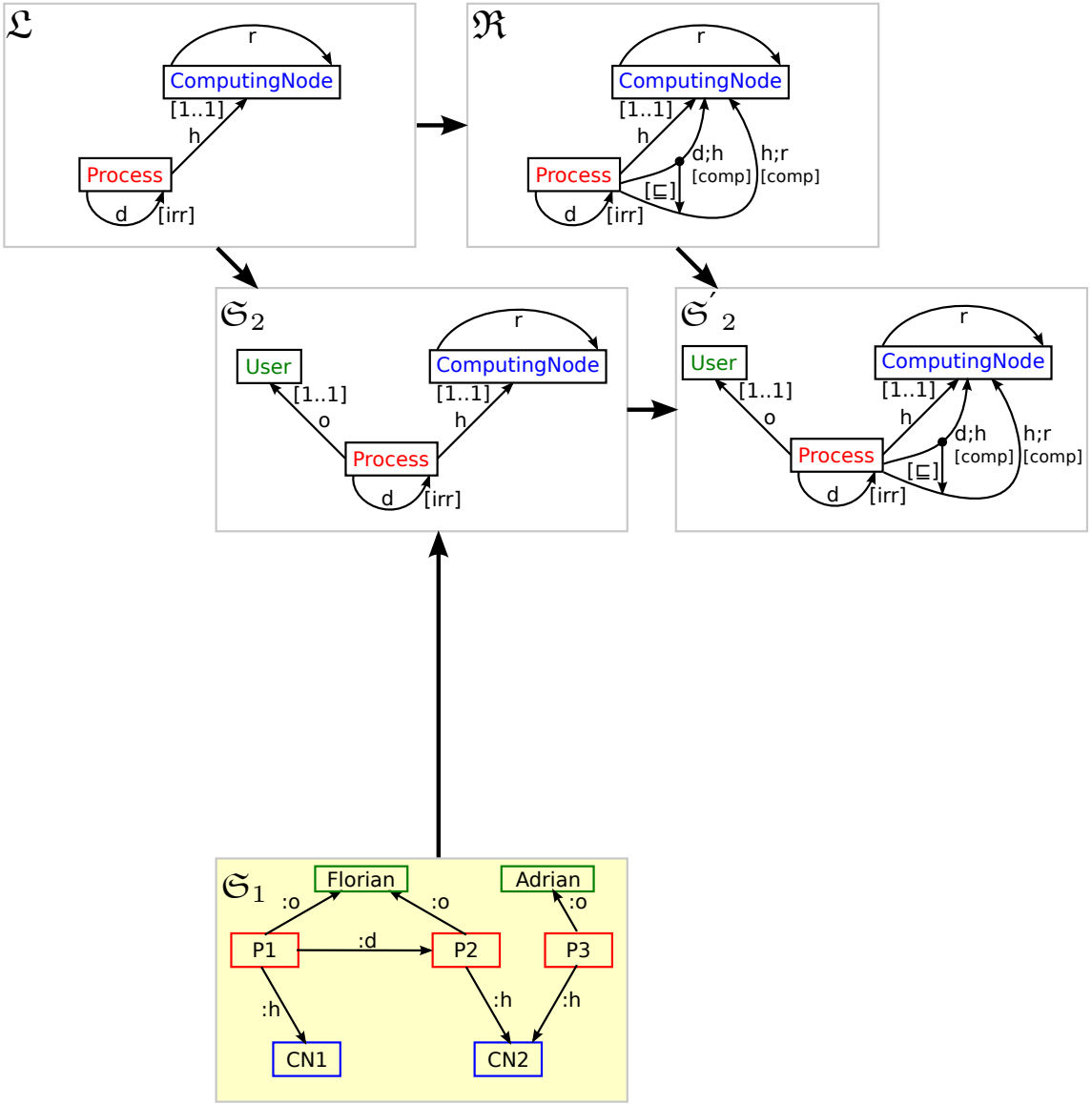
Example



Metamodel Evolution

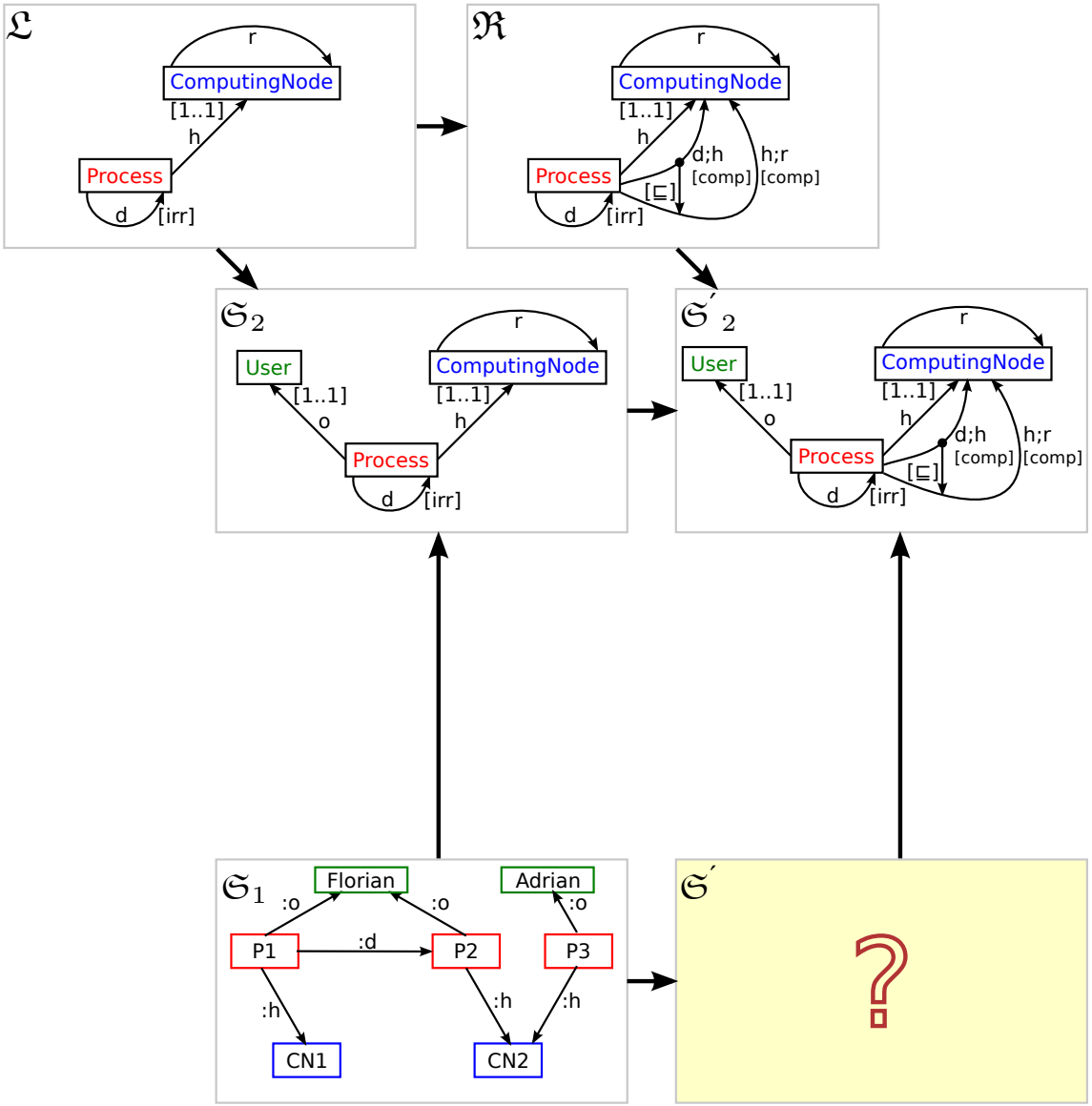
Define metamodel evolution by rule: pushout construction

Example



For any model $\mathfrak{S}_1$ conforming to the metamodel $\mathfrak{S}_2$

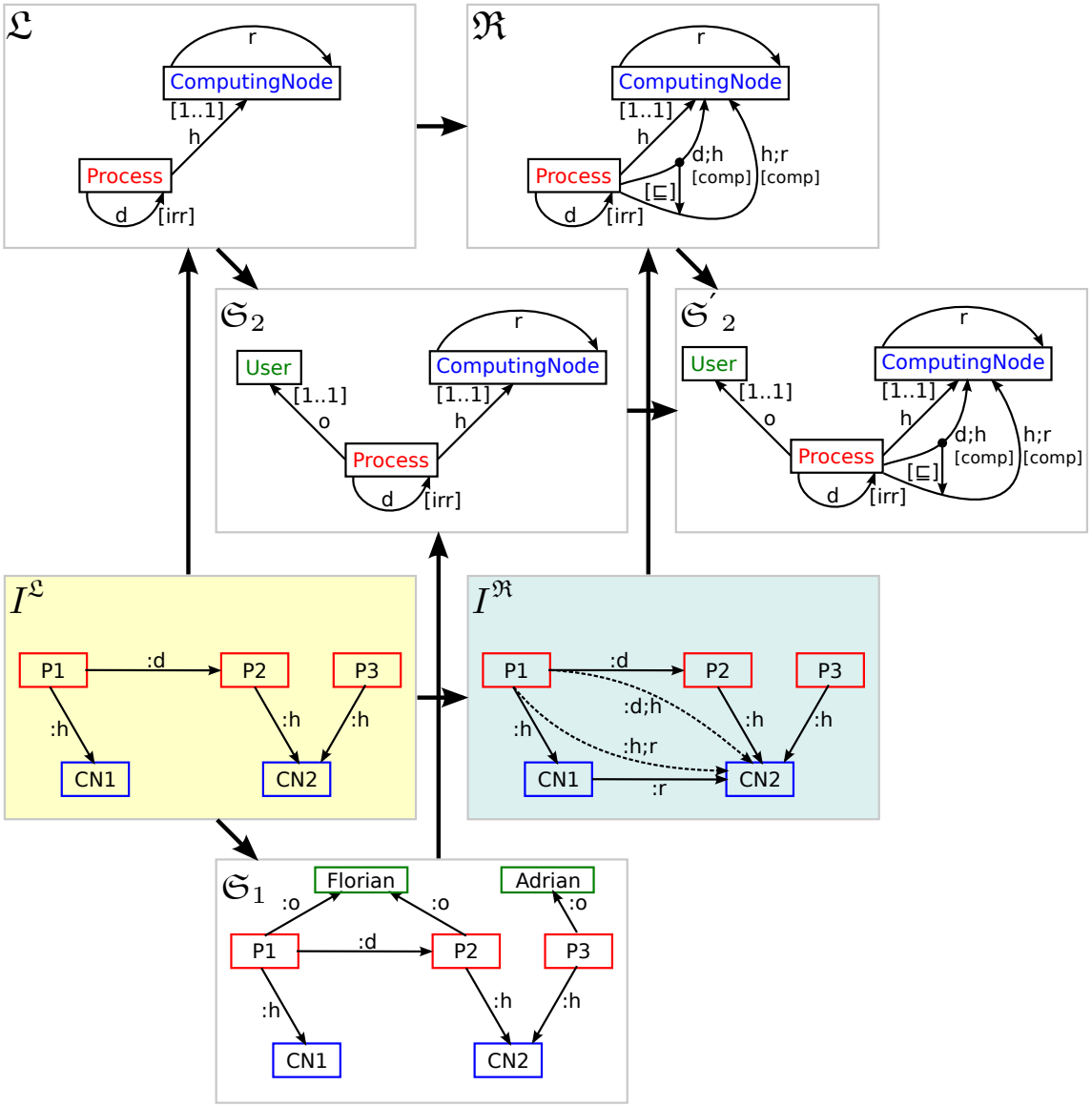
Example



For any model $\mathfrak{S}_1$ conforming to the metamodel $\mathfrak{S}_2$

What is the corresponding model $\mathfrak{S}'_1$ conforming to the metamodel $\mathfrak{S}'_2$?

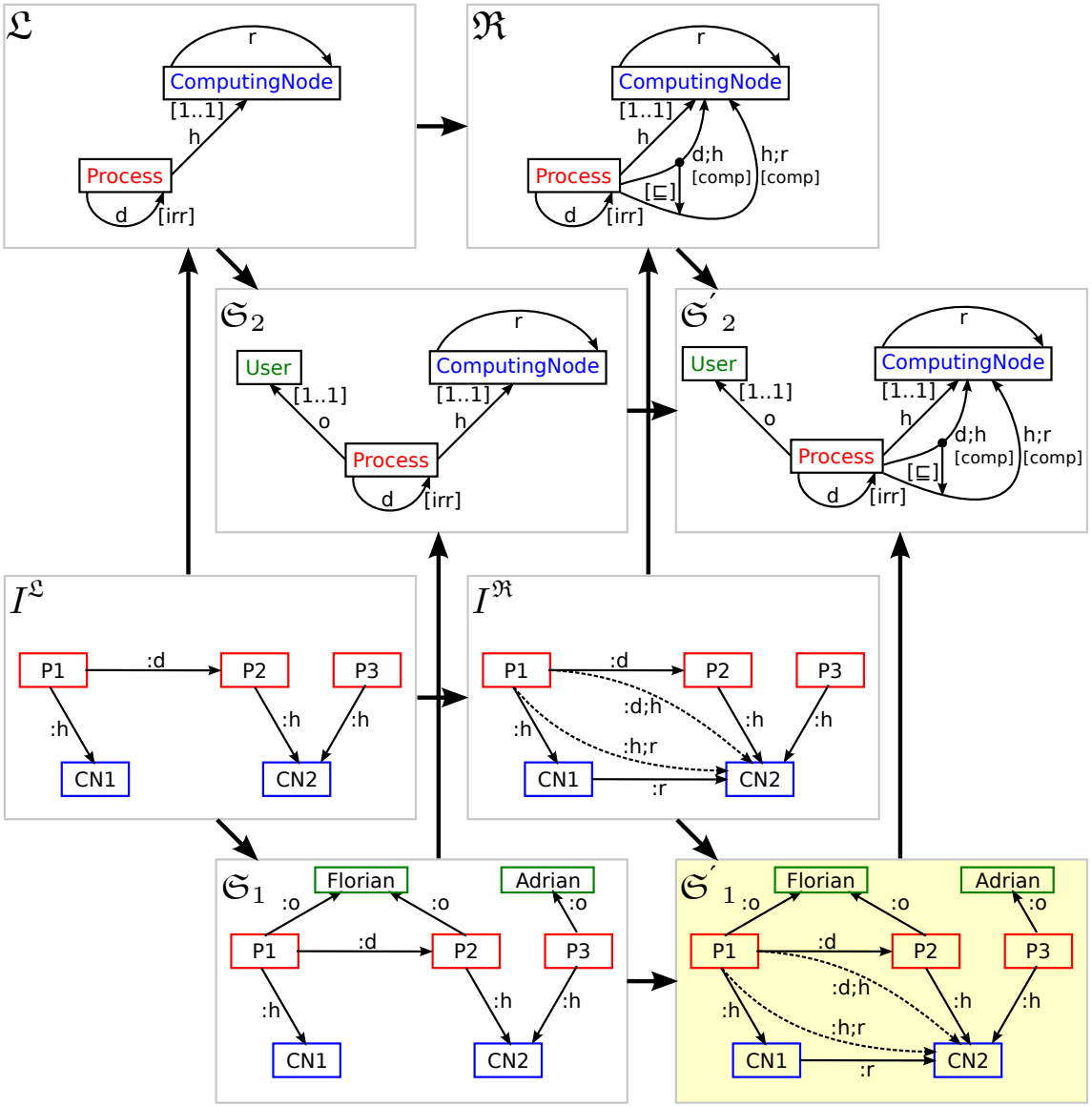
Example



$I^{\mathfrak{L}}$ can be calculated by pullback construction

$I^{\mathfrak{R}}$ can be constructed as ***a*** pullback complement

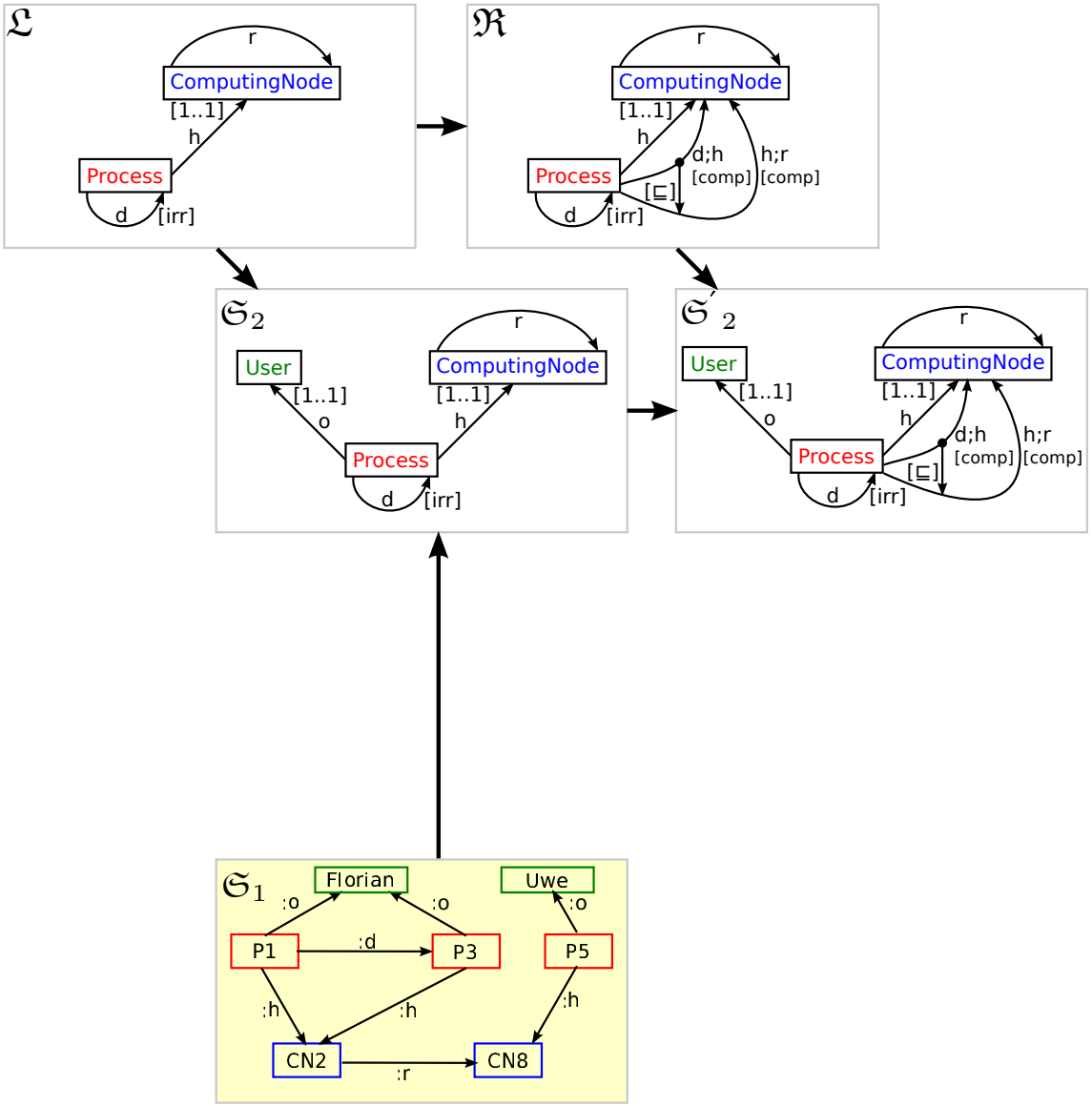
Example



Migration rule $I^{\mathfrak{L}} \to I^{\mathfrak{R}}$ applied to the model $\mathfrak{S}_1$ must create a commuting cube

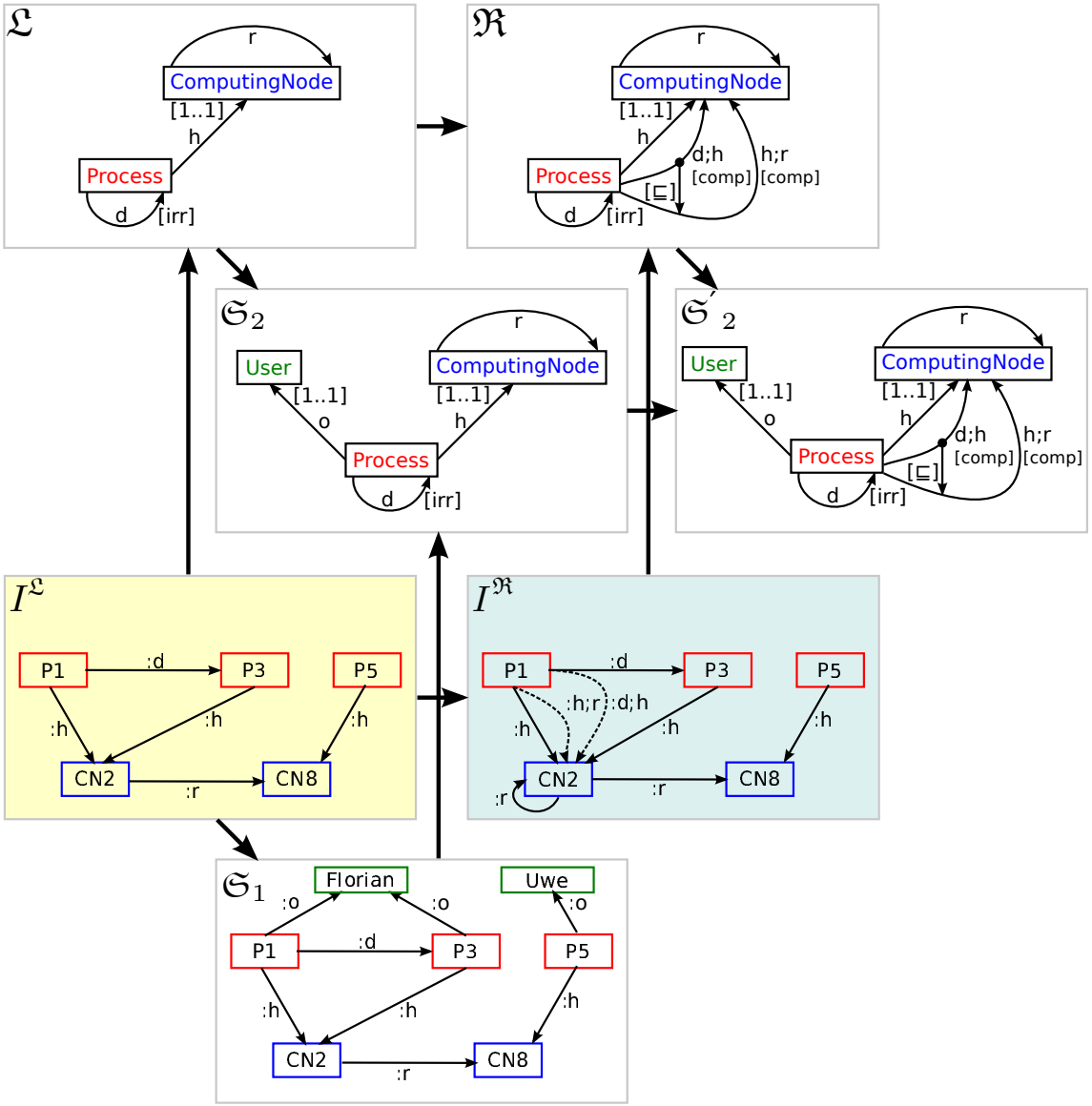
There may be other models conforming to the metamodel $\mathfrak{S}_2$

Example



The same metamodel $\mathfrak{S}_2$ and a different model $\mathfrak{S}_1$

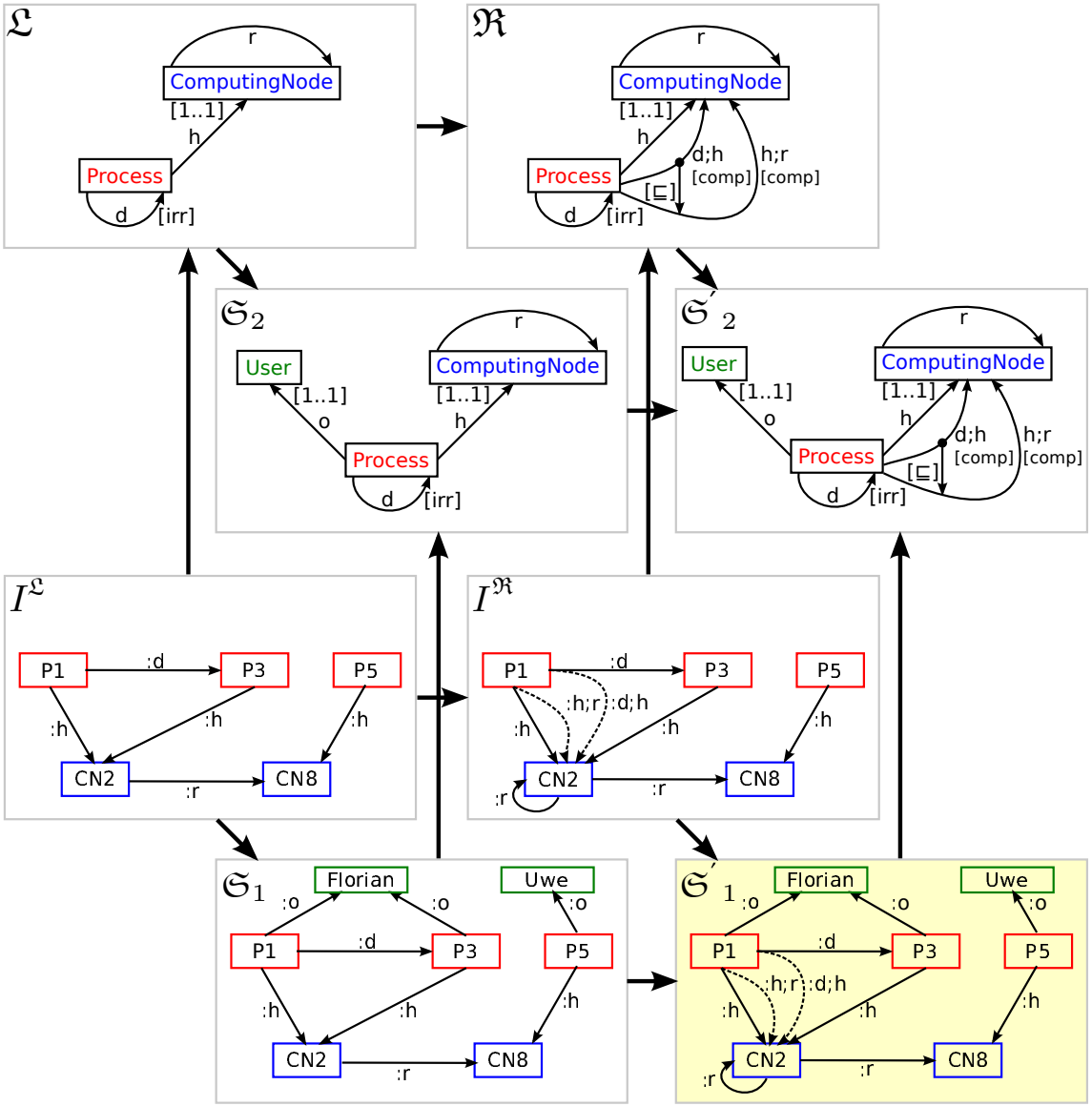
Example



$I^{\mathfrak{L}}$ can be calculated by pullback construction

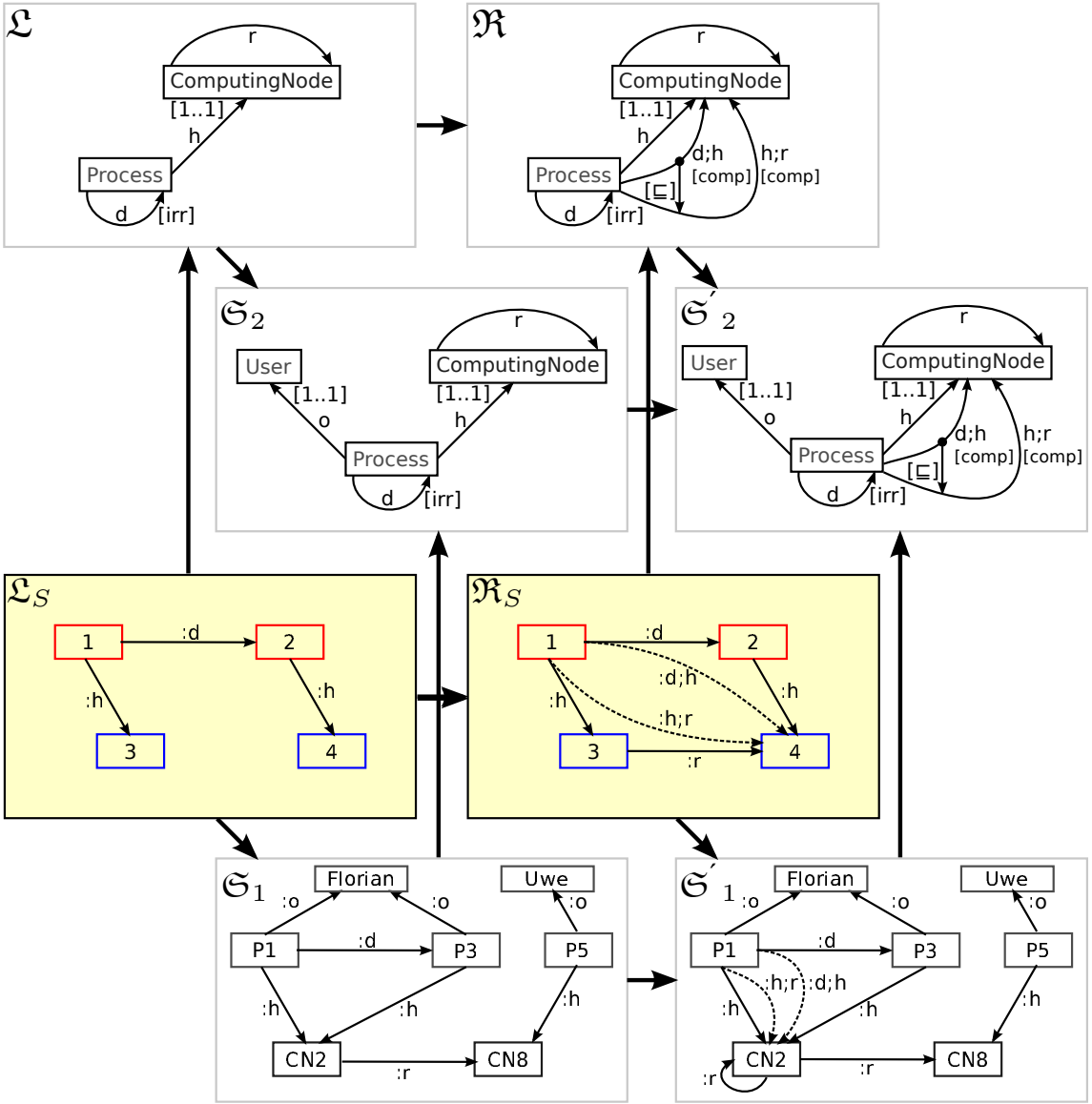
$I^{\mathfrak{R}}$ can be constructed as ***a*** pullback complement

Example



Migration rule $I^{\mathfrak{L}} \rightarrow I^{\mathfrak{R}}$ applied to the model $\mathfrak{S}_1$ must create again a commuting cube

Example



Need an equivalent rule that can migrate all models

Communting Cube

$$\begin{tikzcd}
\mathfrak{L} \arrow[rr, "r"] \arrow[dr, "m"'] & & \mathfrak{R} \arrow[dr, "m'"] & \\
& \mathfrak{S}_2 \arrow[rr, "r'"] & & \mathfrak{S}'_2
\end{tikzcd}$$

Communting Cube

$$\begin{tikzcd}
\mathfrak{L} \arrow[rr, "r"] \arrow[dr, "m"'] & & \mathfrak{R} \arrow[dr, "m'"] & \\
& \mathfrak{S}_2 \arrow[rr, "r'"] & & \mathfrak{S}'_2
\end{tikzcd}$$

- The top face is a pushout

Communting Cube

$$
\begin{tikzcd}
\mathfrak{L} \arrow[rr, "r"] \arrow[dr, "m"] & & \mathfrak{R} \arrow[dr, "m'"] & \\
& \mathfrak{S}_2 \arrow[rr, "r'"] & & \mathfrak{S}'_2 \\
I^{\mathfrak{L}} \arrow[uu, "\iota^{\mathfrak{L}}"] \arrow[dr, "m^*"] & & & \\
& \mathfrak{S}_1 \arrow[uu, "\iota^{S_1}"] & &
\end{tikzcd}
$$

- The left face is a pullback

Communting Cube

$$
\begin{tikzcd}
\mathfrak{L} \arrow[rr, "r"] \arrow[dr, "m"] & & \mathfrak{R} \arrow[dr, "m'"] & \\
& \mathfrak{S}_2 \arrow[rr, "r'"] & & \mathfrak{S}'_2 \\
I^{\mathfrak{L}} \arrow[uu, "\iota^{\mathfrak{L}}"] \arrow[rr, "r^*"] \arrow[dr, "m^*"] & & I^{\mathfrak{R}} \arrow[uu, "[\![r]\!](\iota^{\mathfrak{R}})"'] & \\
& \mathfrak{S}_1 \arrow[uu, "\iota^{S_1}"] & &
\end{tikzcd}
$$

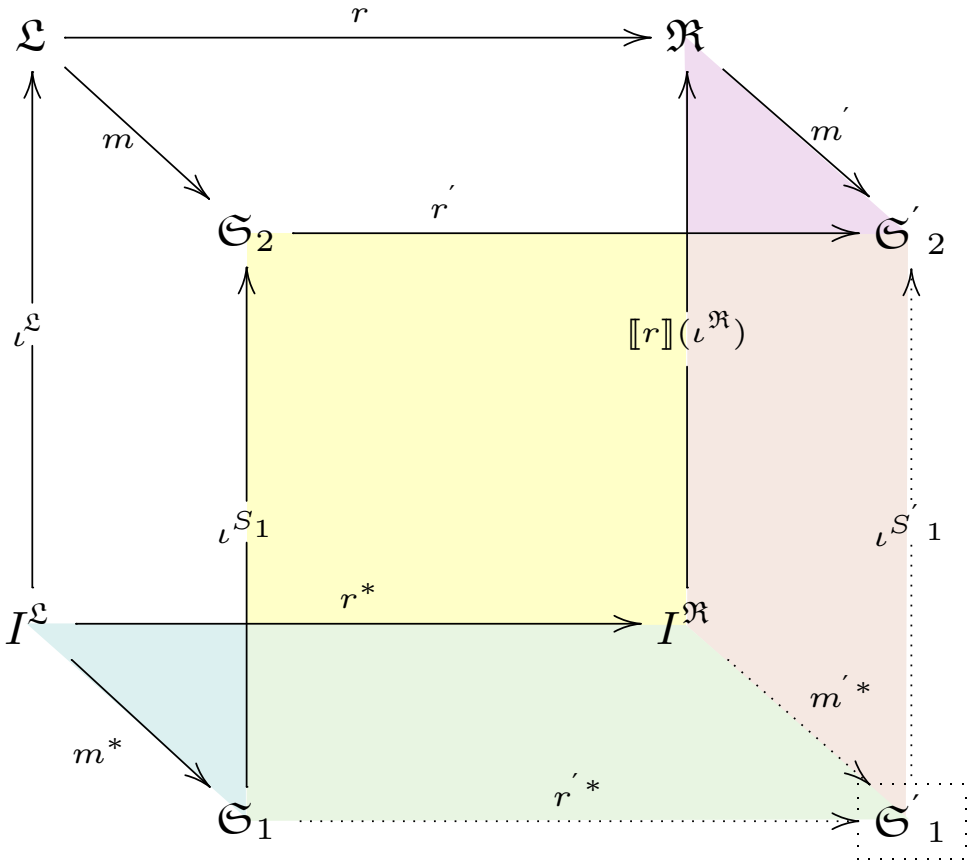
- The back face is a pullback

Communting Cube

$$
\begin{tikzcd}
\mathfrak{L} \arrow[rr, "r"] \arrow[dr, "m"] & & \mathfrak{R} \arrow[dr, "m'"] & \\
& \mathfrak{S}_2 \arrow[rr, "r'"] & & \mathfrak{S}'_2 \\
I^{\mathfrak{L}} \arrow[uu, "\iota^{\mathfrak{L}}"] \arrow[rr, "r^*"] \arrow[dr, "m^*"] & & I^{\mathfrak{R}} \arrow[uu, "[\![r]\!](\iota^{\mathfrak{R}})"'] & \\
& \mathfrak{S}_1 \arrow[uu, "\iota^{S_1}"] & &
\end{tikzcd}
$$

- For a rule r the semantics $[\![r]\!]$ can be any mapping satisfying the condition that the back face is a pullback

Communting Cube



- If the bottom face is a pushout
- Then the front face and the right face have to be pullbacks

Communting Cube

$$
\begin{tikzcd}
\mathfrak{L} \arrow[rr, "r"] & & \mathfrak{R} \\
I^{\mathfrak{L}} \arrow[u, "\iota^{\mathfrak{L}}"] \arrow[rr, "r^*"] \arrow[dr, "m^*"'] & & I^{\mathfrak{R}} \arrow[u, "[\![r]\!](\iota^{\mathfrak{R}})"'] \arrow[dr, dotted, "m'^*"] & \\
& \mathfrak{S}_1 \arrow[rr, dotted, "r'^*"] & & \boxed{\mathfrak{S}'_1}
\end{tikzcd}
$$

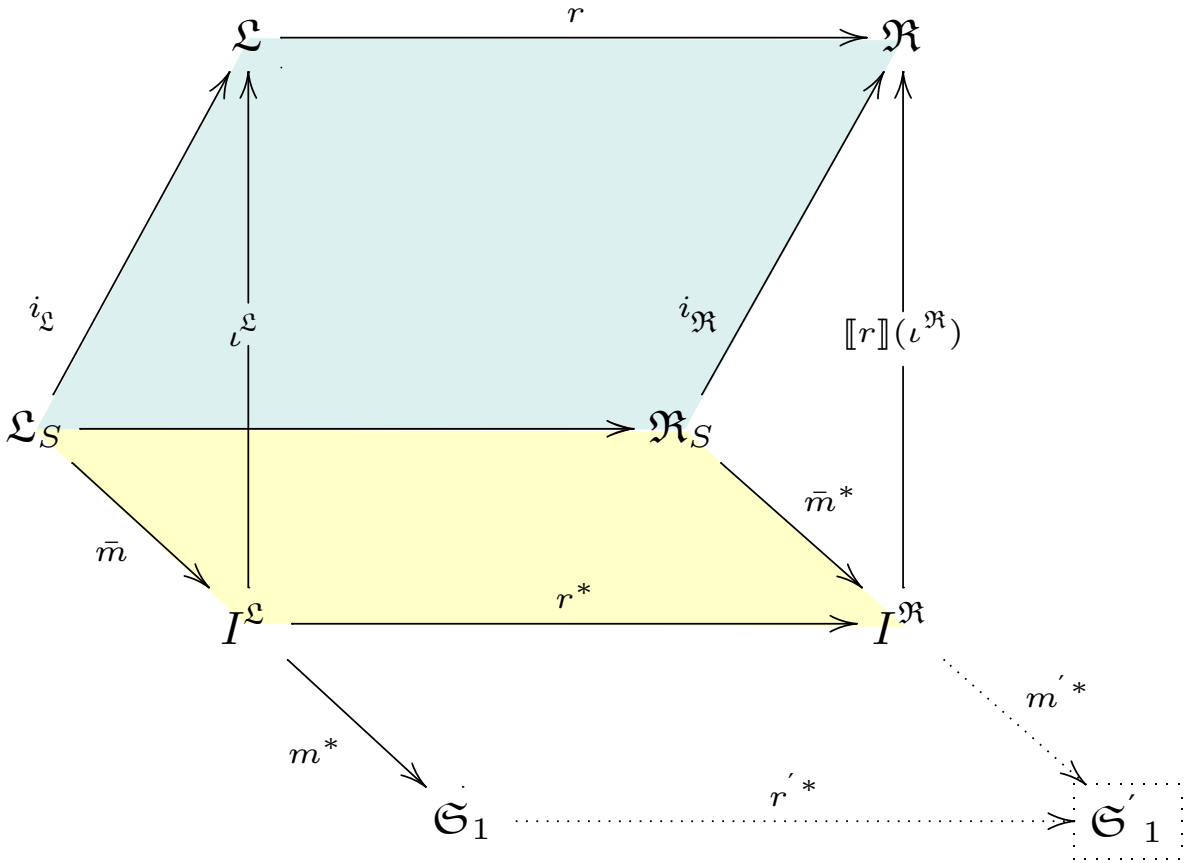
- Need for a rule which captures all conforming models

Communting Cube

$$
\begin{tikzcd}
 & \mathfrak{L} \arrow[rr, "r"] & & \mathfrak{R} \\
\mathfrak{L}_S \arrow[ur, "i_{\mathfrak{L}}"] \arrow[rr, "r_s"] \arrow[dr, "\bar{m}"'] & & \mathfrak{R}_S \arrow[ur, "i_{\mathfrak{R}}"] \arrow[dr, "\bar{m}^*"] & \\
 & I^{\mathfrak{L}} \arrow[uu, "\iota^{\mathfrak{L}}"] \arrow[rr, "r^*"] \arrow[dr, "m^*"'] & & I^{\mathfrak{R}} \arrow[uu, "[\![r]\!](\iota^{\mathfrak{R}})"'] \arrow[dr, dotted, "m'^*"] \\
 & & \mathfrak{S}_1 \arrow[rr, dotted, "r'^*"] & & \boxed{\mathfrak{S}'_1}
\end{tikzcd}
$$

- The rule r_s has to be minimal
- It has to complete the cube for any conforming model $\mathfrak{S}_1$

Communting Cube



- The blue parallelogram has to commute
- The yellow parallelogram has to be a pushout

Future Work

- Adapt the DPF approach to constraint-aware model transformation for all layers of a metamodelling hierarchy
- Find the conditions on the rules that guarantee a commuting cube
- Extend the cube for the case that one minimal rule is not sufficient
- Investigate constraints on the model level
- Investigate evolution of model transformations

Thank you!

Questions?

For more information and list of publications visit:
`http://dpf.hib.no/`