Constraint-Aware Model Transformations

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

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

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

16 October 2009
NWPT 2009, Lyngby, Denmark

HØGSKOLEN I BERGEN

Outline

MDE and model transformation

- Generation of target models from source models
- Model Transformation (MT) plays a central role in MDE
- Application areas in MDE:
  - Development process: code generation, refinement etc
  - Model management: integration, decomposition etc
  - Migration: from a platform, an implementation technology etc, to another
  - Language translation: from a modelling language to another

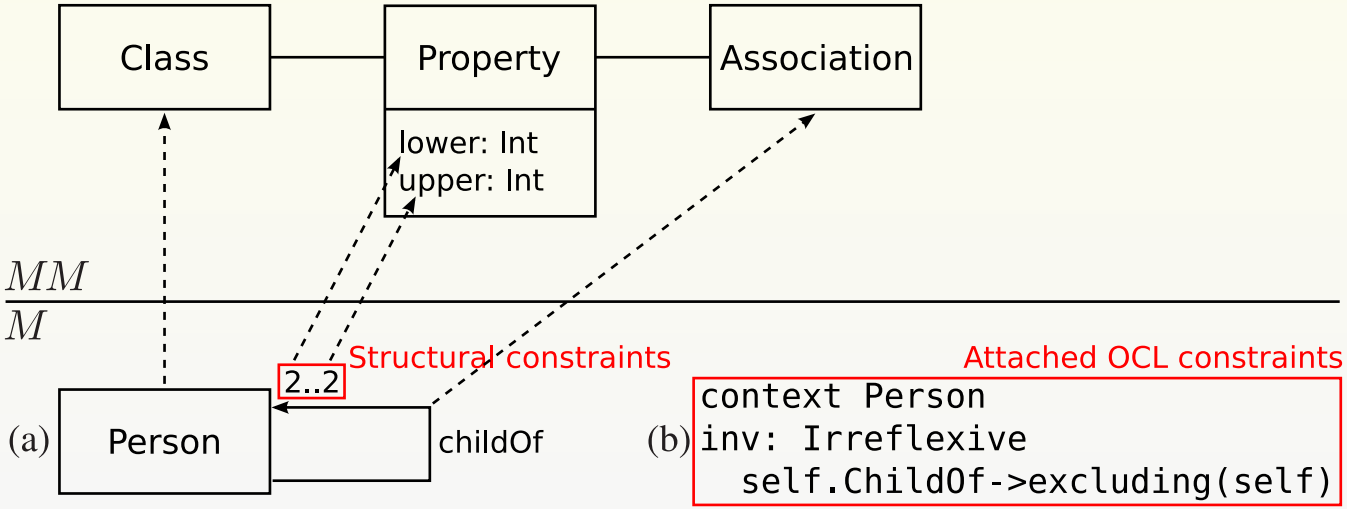
What is model transformation?

- L_S, L_T source and target modelling languages
  - Usually graph-based languages e.g. UML or EMF
  - Have a corresponding metamodel MM_S, MM_T
- S, T source and target models
  - Specified by the modelling languges
  - Conform to the metamodel of the languages

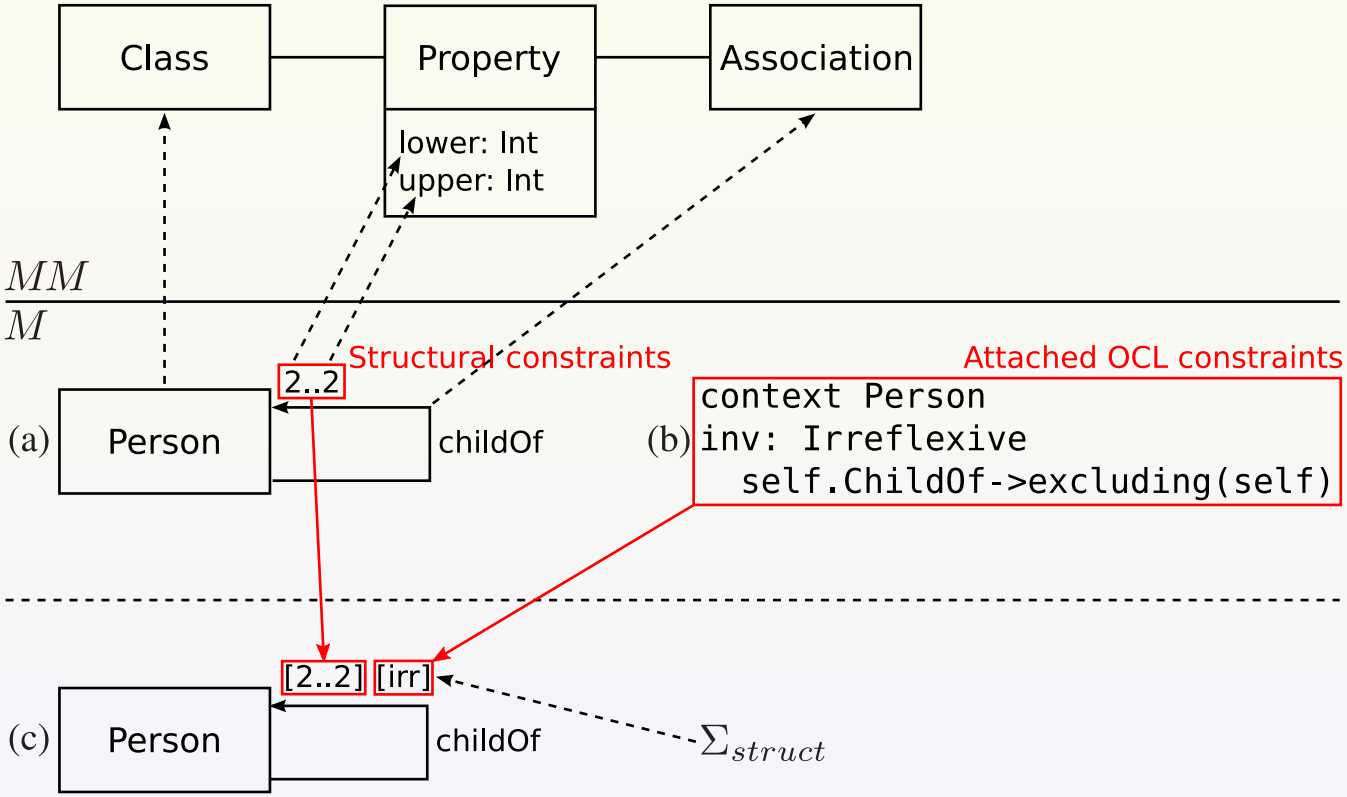
$$\begin{tikzcd}
MM_S \arrow[r, "rules"] & MM_T \\
S \arrow[u, "conforms"] \arrow[r, Rightarrow, "MT"] & T \arrow[u, "conforms"']
\end{tikzcd}$$

Challenge

Challenge


Class
Property
Association
lower: Int
upper: Int
MM
M
Structural constraints
2..2
(a)
Person
childOf
Attached OCL constraints
(b)
context Person
inv: Irreflexive
self.ChildOf->excluding(self)

Challenge


Class
Property
Association
lower: Int
upper: Int
MM
M
Structural constraints
Attached OCL constraints
2..2
(a)
Person
childOf
(b)
context Person
inv: Irreflexive
self.ChildOf->excluding(self)
[2..2]
[irr]
(c)
Person
childOf
Σ_{struct}

Our contribution

- Diagrammatic modelling framework
  - Integration of constraints in modelling
- Enables support for specification of constraint-aware rules

Modelling formalisms

- Modelling language $\xrightarrow{formalisation}$ modelling formalism in DPF
- A modelling formalism $(\Sigma_1, S_2, \Sigma_2)$

$$\begin{tikzcd}
\Sigma_2 \arrow[r, dashed, "C^{S_2}"] & G^{S_2} \\
\Sigma_1 \arrow[r, dashed, "C^{S_1}"] & G^{S_1} \arrow[u, "\iota^{S_1}"']
\end{tikzcd}$$

Constraint-aware model transformation

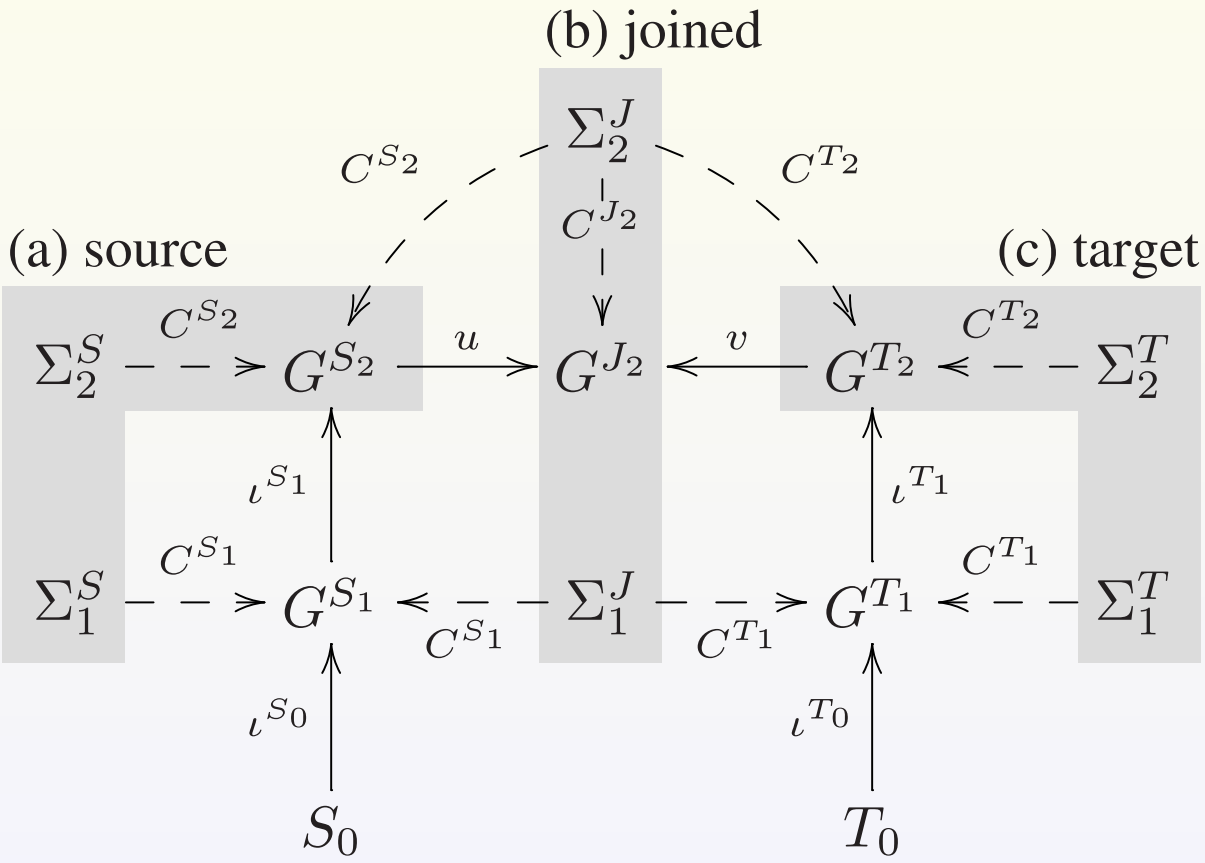
(a) source

$$\begin{tikzcd}
\Sigma_2^S \arrow[r, dashed, "C^{S_2}"] & G^{S_2} \\
\Sigma_1^S \arrow[r, dashed, "C^{S_1}"] & G^{S_1} \arrow[u, "\iota^{S_1}"] \\
 & S_0 \arrow[u, "\iota^{S_0}"]
\end{tikzcd}$$

(c) target

$$\begin{tikzcd}
G^{T_2} & \Sigma_2^T \arrow[l, dashed, "C^{T_2}"'] \\
G^{T_1} \arrow[u, "\iota^{T_1}"] & \Sigma_1^T \arrow[l, dashed, "C^{T_1}"'] \\
T_0 \arrow[u, "\iota^{T_0}"] &
\end{tikzcd}$$

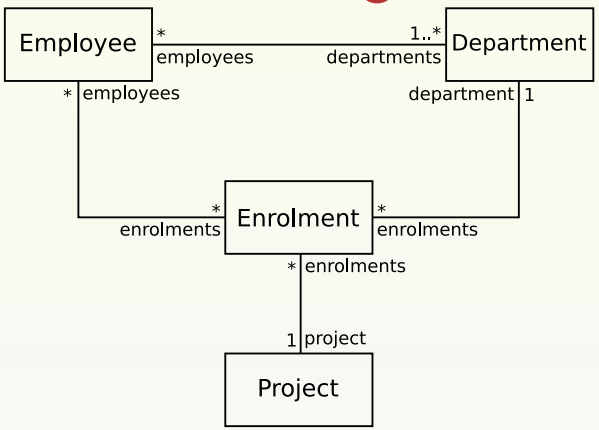
Constraint-aware model transformation



Outline

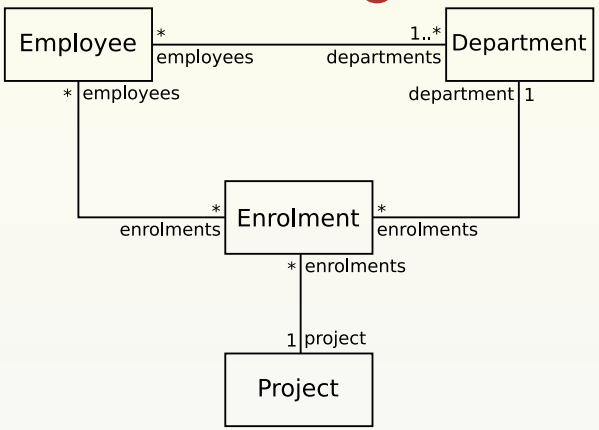
Recall the motivating example

UML class diagram



Recall the motivating example

UML class diagram

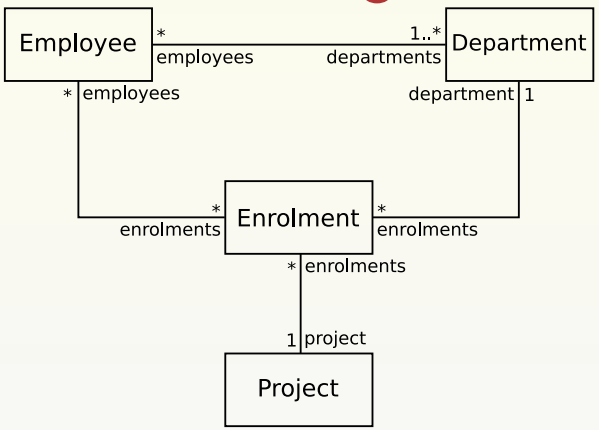


OCL

```
context Enrolment
 inv rule6: self.department.employees ->
  includesAll(self.employee)
 inv rule7: Let enrolments:Set(Enrolment)=
   Enrolment.allInstances in
   (not enrolment->exists(enr|
    enr.project=self.project
    and enr.department=self.department
    and enr.employees=self.employees))
```

Recall the motivating example

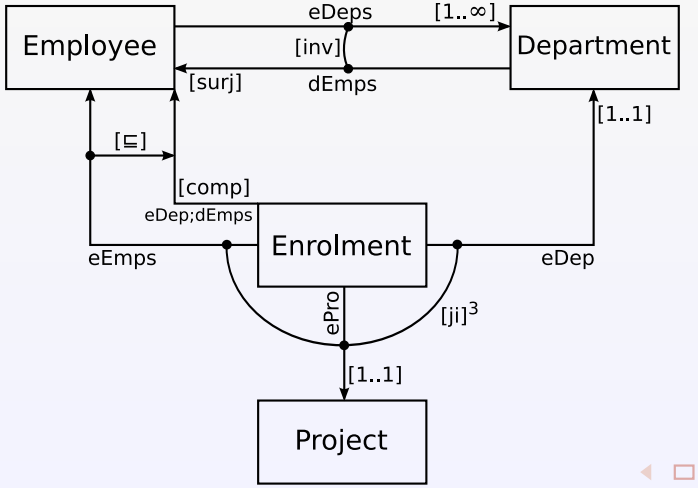
UML class diagram



OCL

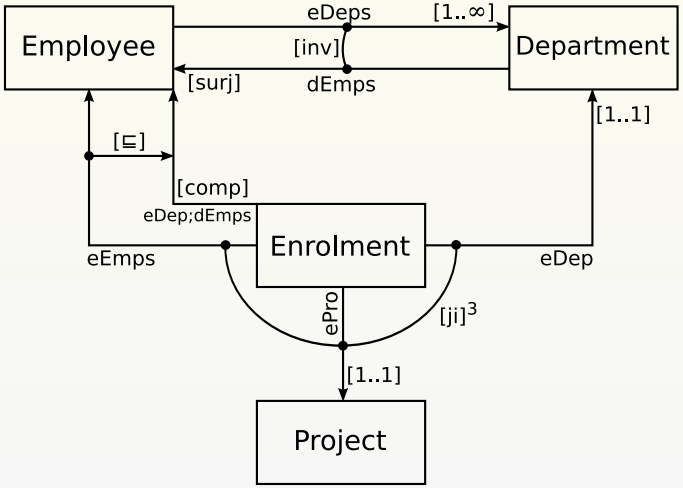
```
context Enrolment
 inv rule6: self.department.employees ->
  includesAll(self.employee)
 inv rule7: Let enrolments:Set(Enrolment)=
   Enrolment.allInstances in
   (not enrolment->exists(enr|
    enr.project=self.project
    and enr.department=self.department
    and enr.employees=self.employees))
```

DPF



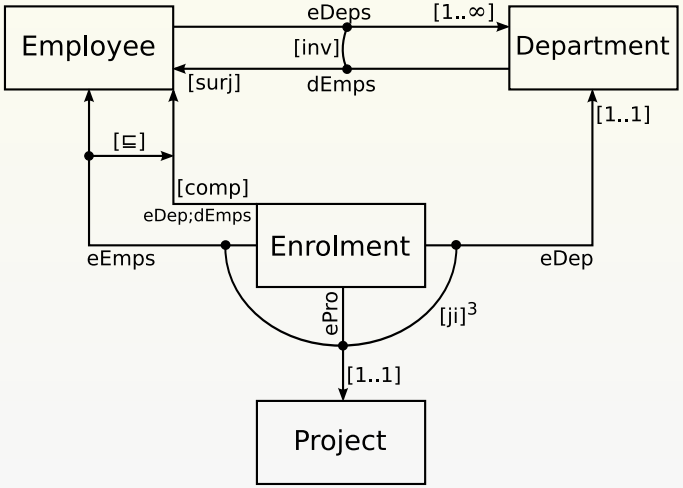
Recall the motivating example

Source: structural model (Σ_{struct}-specification)

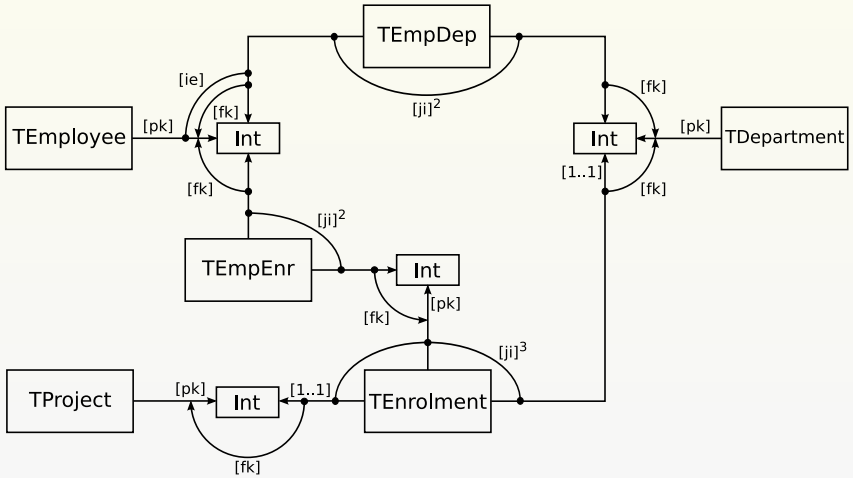


Recall the motivating example

Source: structural model (Σ_{struct}-specification)



Target: relational model (Σ_{rel}-specification)



Signature Σ_{rel}

p	$\alpha_{rel}(p)$	Proposed visualisat.	Intended semantics
[primary-key]	$1 \xrightarrow{f} 2$	$\boxed{X} \xrightarrow[\text{[pk]}]{f} \boxed{Y}$	f is [mult(1,1)], [injective]
[foreign-key]	$\begin{tikzcd} 1 \arrow[r, "f"] & 2 \\ & 3 \arrow[u, "g"] \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, "f"] \arrow[dr, bend right, "\text{[fk]}"'] & \boxed{Y} \\ & \boxed{Z} \arrow[u, "g"'] \end{tikzcd}$	$f(X) \subseteq g(Y)$
[image-equal]	$\begin{tikzcd} 1 \arrow[r, "f"] & 2 \\ & 3 \arrow[u, "g"] \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, "f"] \arrow[dr, bend right, "\text{[ie]}"'] & \boxed{Y} \\ & \boxed{Z} \arrow[u, "g"'] \end{tikzcd}$	$f(X) = g(Z)$

Outline

Steps

1. Join the modelling formalisms
2. Define transformation rules
3. Extend S to be typed by the joined metamodel
4. Apply the transformation rules
5. Projection of T

Joined modelling formalism

- Create $(\Sigma_1^J, J_2, \Sigma_2^J)$
- $\Sigma_1^J := \Sigma_1^S \uplus \Sigma_1^T$ and $\Sigma_2^J := \Sigma_2^S \uplus \Sigma_2^U \uplus \Sigma_2^T$
- $J_2 := S_2 \uplus U_2 \uplus T_2$

(a) source

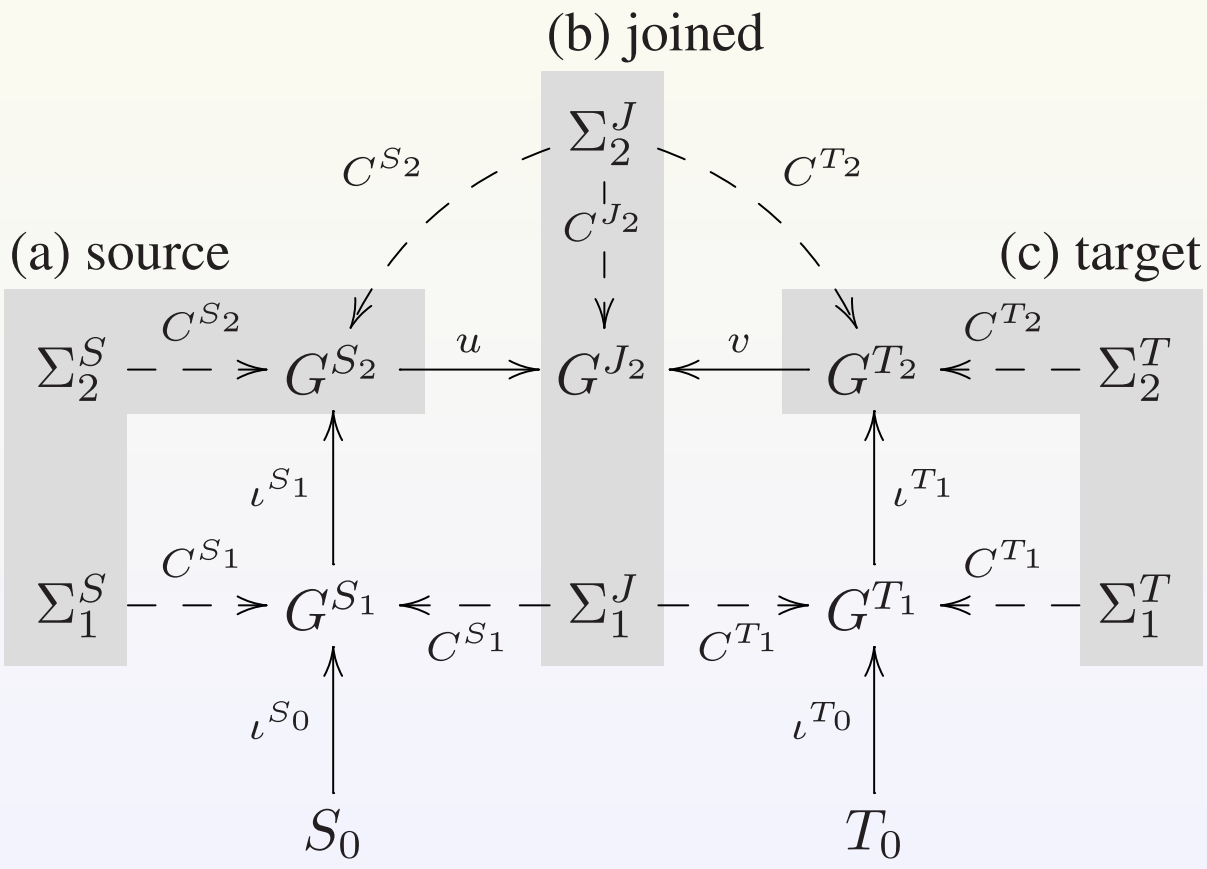
$$\begin{tikzcd}
\Sigma_2^S \arrow[r, dashed, "C^{S_2}"] & G^{S_2} \\
\Sigma_1^S \arrow[r, dashed, "C^{S_1}"] & G^{S_1} \arrow[u, "\iota^{S_1}"] \\
 & S_0 \arrow[u, "\iota^{S_0}"]
\end{tikzcd}$$

(c) target

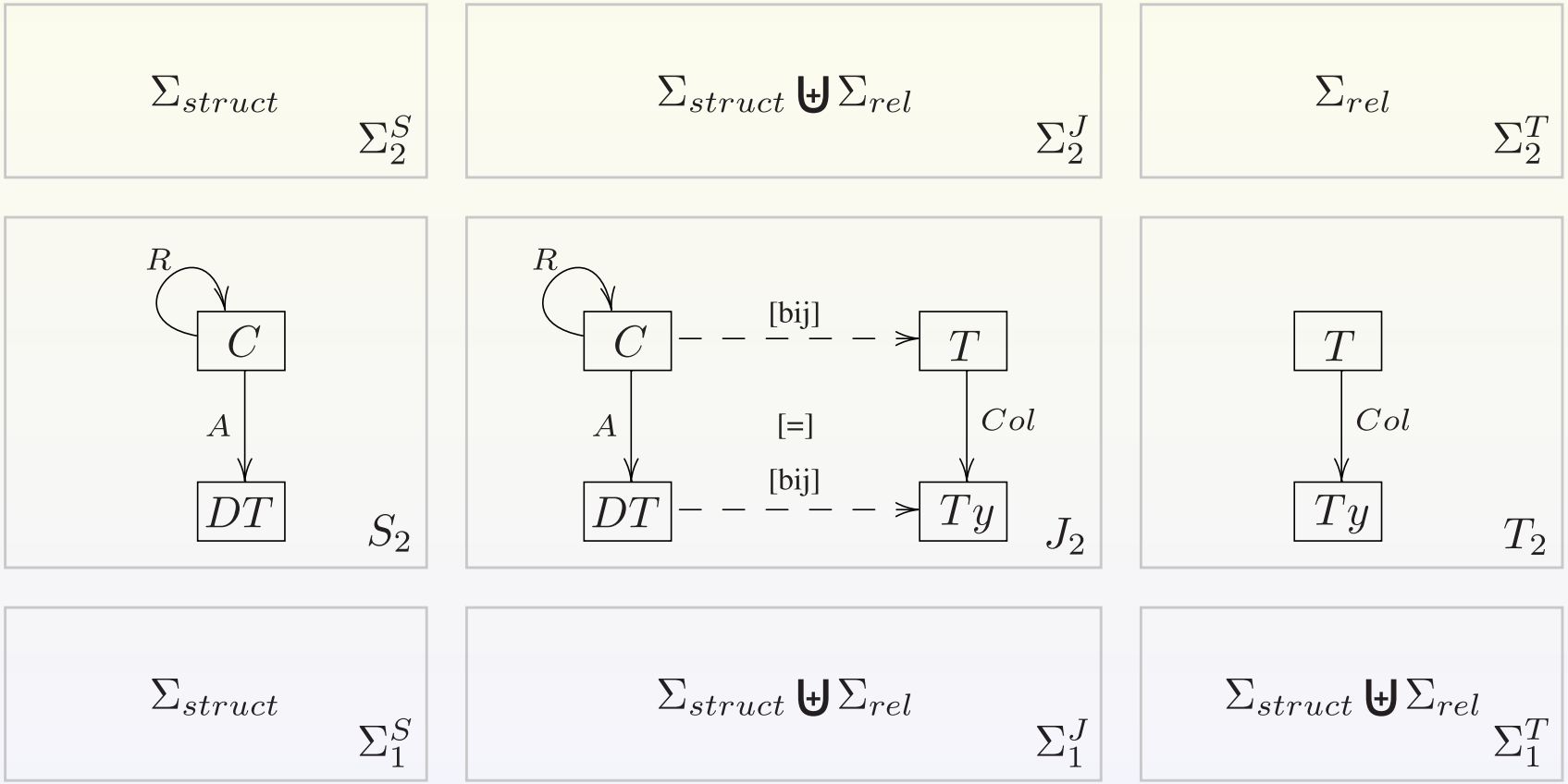
$$\begin{tikzcd}
G^{T_2} & \Sigma_2^T \arrow[l, dashed, "C^{T_2}"'] \\
G^{T_1} \arrow[u, "\iota^{T_1}"] & \Sigma_1^T \arrow[l, dashed, "C^{T_1}"'] \\
T_0 \arrow[u, "\iota^{T_0}"]
\end{tikzcd}$$

Joined modelling formalism

- Create $(\Sigma_1^J, J_2, \Sigma_2^J)$
- $\Sigma_1^J := \Sigma_1^S \uplus \Sigma_1^T$ and $\Sigma_2^J := \Sigma_2^S \uplus \Sigma_2^U \uplus \Sigma_2^T$
- $J_2 := S_2 \uplus U_2 \uplus T_2$



Example of join



Signature Σ_{join}

[commutative]	$\begin{tikzcd} 1 \arrow[r, "f"] \arrow[d, "g"] & 2 \arrow[d, "y'"] \\ 3 \arrow[r, "x'"] & 4 \end{tikzcd}$	$\begin{tikzcd} \boxed{X} \arrow[r, "f"] \arrow[d, "g"] \arrow[dr, phantom, "[=]"] & \boxed{Y} \arrow[d, "g'"] \\ \boxed{Z} \arrow[r, "f'"] & \boxed{Æ} \end{tikzcd}$	$\forall x \in X : g'(f(x)) = f'(g(x))$

Projection condition

$$\begin{tikzcd} G^{S_2} \arrow[r, hook, "u"] & G^{J_2} & G^{T_2} \arrow[l, hook', "v"'] \\ & G^{J_1} \arrow[u, dashed, "\iota^{J_1}"] & \end{tikzcd}$$

- `[bijective]` on arrows in G^{U_2}

Projection condition

$$
\begin{tikzcd}
G^{S_2} \arrow[r, hook, "u"] & G^{J_2} & G^{T_2} \arrow[l, hook', "v"'] \\
G^{S_1} \arrow[u, "\iota^{S_1}"] \arrow[r, hook, "\pi^{S_1}"'] \arrow[ur, phantom, "P.B."] & G^{J_1} \arrow[u, "\iota^{J_1}"] & G^{T_1} \arrow[u, "\iota^{T_1}"] \arrow[l, hook', "\pi^{T_1}"] \arrow[ul, phantom, "P.B."]
\end{tikzcd}
$$

- `[bijective]` on arrows in G^{U_2}

Define constraint-aware rules

$$\begin{tikzcd} & G^{J_2} & \\ L \arrow[ur, "t_L"] \arrow[rr, hook, "r"] & = & R \arrow[ul, "t_R"'] \end{tikzcd}$$

- Both L and R are G^{J_2}-typed Σ_1^J-specifications
- r is a G^{J_2}-typed Σ_1^J-specification morphism
- Application of r via a match m

Specification Morphism

$$\begin{tikzcd} \alpha(p) \arrow[r, "\delta"] \arrow[dr, "\delta;r"'] & G^L \arrow[d, "r"] \\ & G^R \end{tikzcd} \quad =$$

Define constraint-aware rules

$$\begin{tikzcd} & G^{J_2} & \\ L \arrow[ur, "t_L"] \arrow[rr, hook, "r"] & = & R \arrow[ul, "t_R"'] \end{tikzcd}$$

- Both L and R are G^{J_2}-typed Σ_1^J-specifications
- r is a G^{J_2}-typed Σ_1^J-specification morphism
- Application of r via a match m

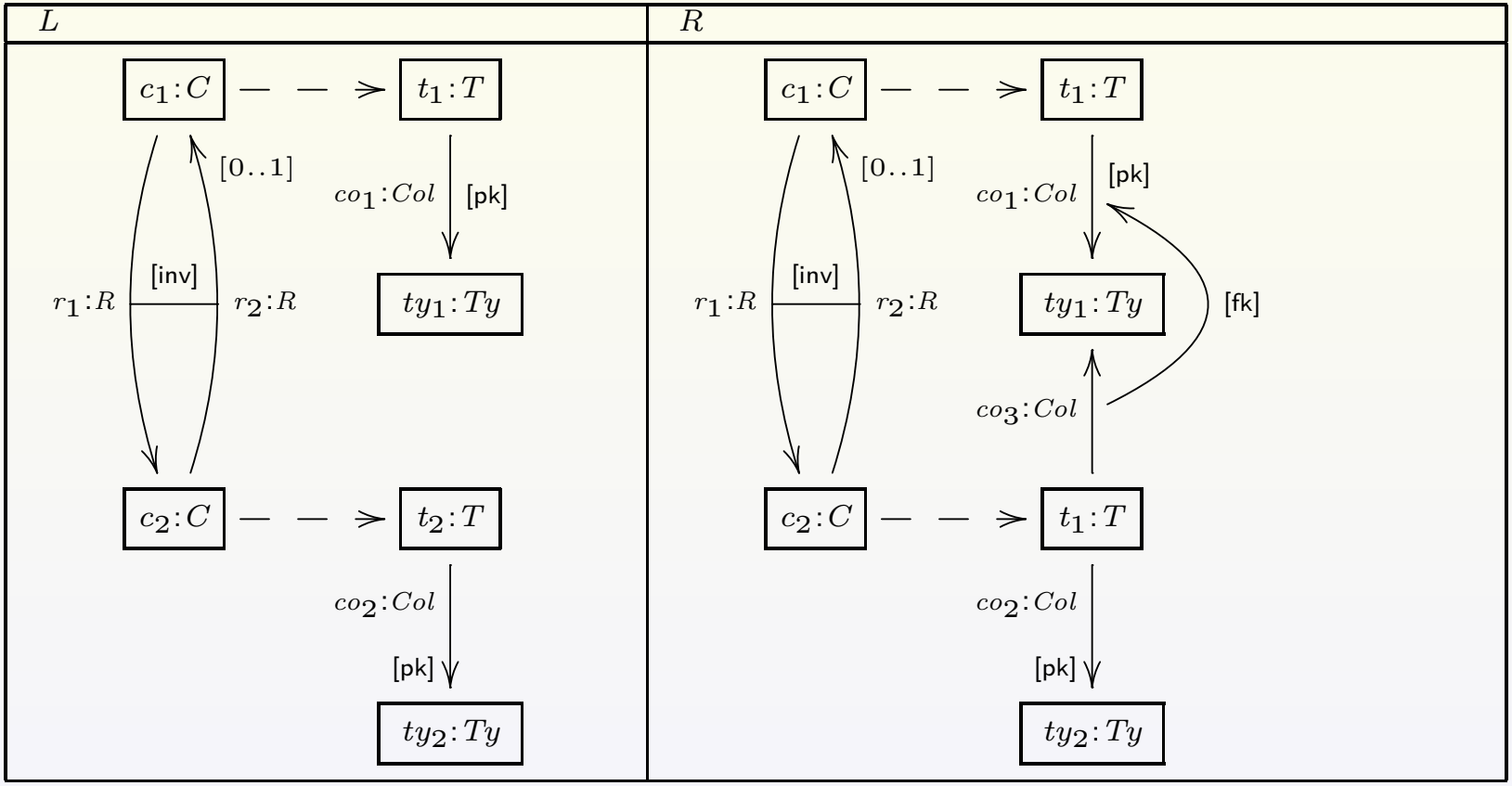
Application of Rule

$$\begin{tikzcd} L \arrow[r, hook, "r"] \arrow[d, "m"'] \arrow[dr, phantom, "P.O."] & R \arrow[d, "m^*"] \\ J_1 \arrow[r, hook, "\langle r,m \rangle"] & J_1' \end{tikzcd}$$

Constraint-aware rules: r_1 and r_2

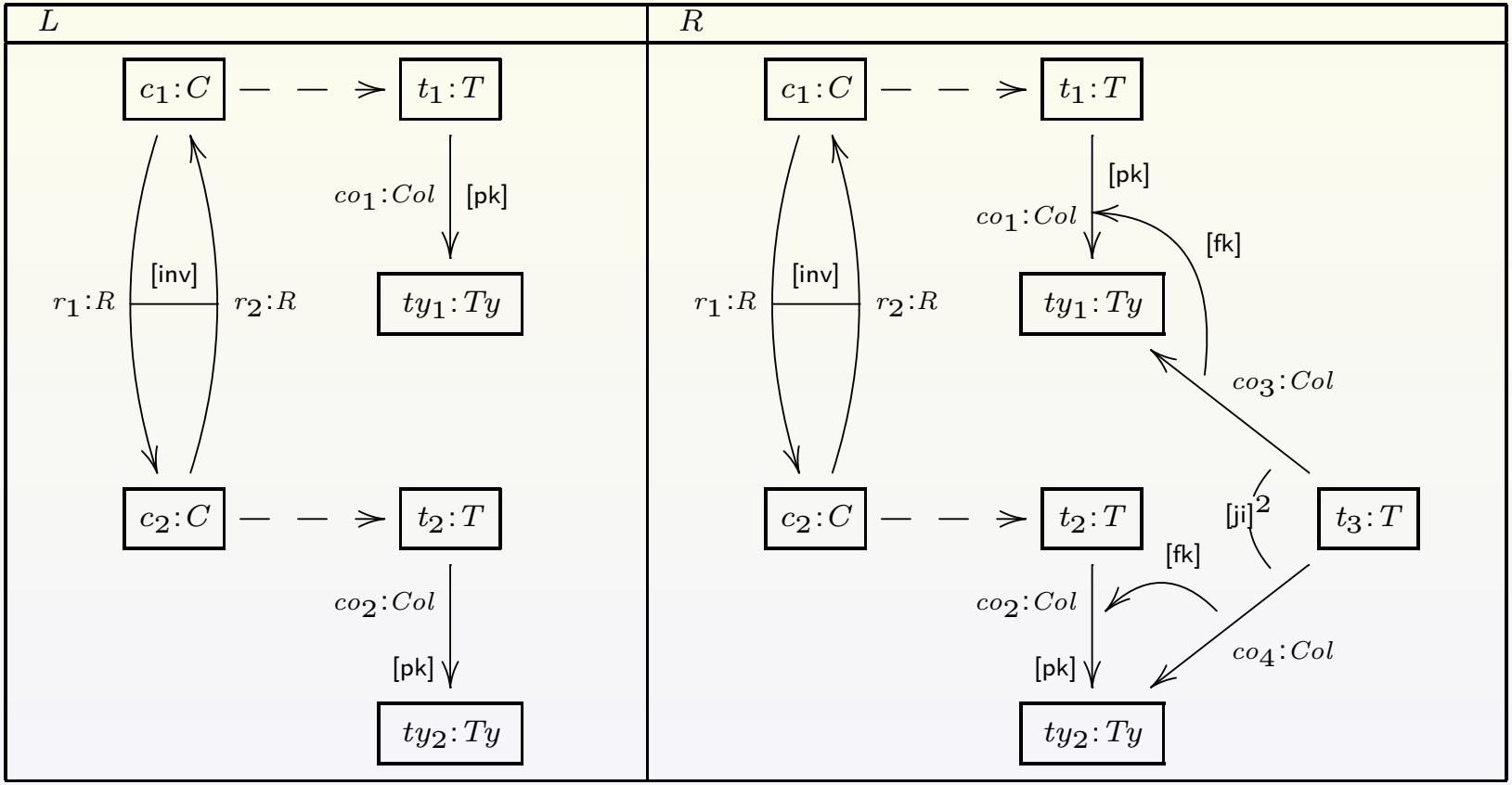
L	R
Rule r_1. Class to table	
$c{:}C$	$c{:}C \dashrightarrow t{:}T \xrightarrow[co{:}Col]{[pk]} ty{:}Ty$
Rule r_2. Attribute to column	
$c{:}C \dashrightarrow t{:}T \xrightarrow[co_1{:}Col]{[pk]} ty_1{:}Ty$ $c{:}C \xrightarrow{a{:}A} dt{:}DT$	$c{:}C \dashrightarrow t{:}T \xrightarrow[co_1{:}Col]{[pk]} ty_1{:}Ty$ $c{:}C \xrightarrow{a{:}A} dt{:}DT$ $t{:}T \xrightarrow{co_2{:}Col} ty_2{:}Ty$ $dt{:}DT \dashrightarrow ty_2{:}Ty$

Rule r_3: Reference to foreign key



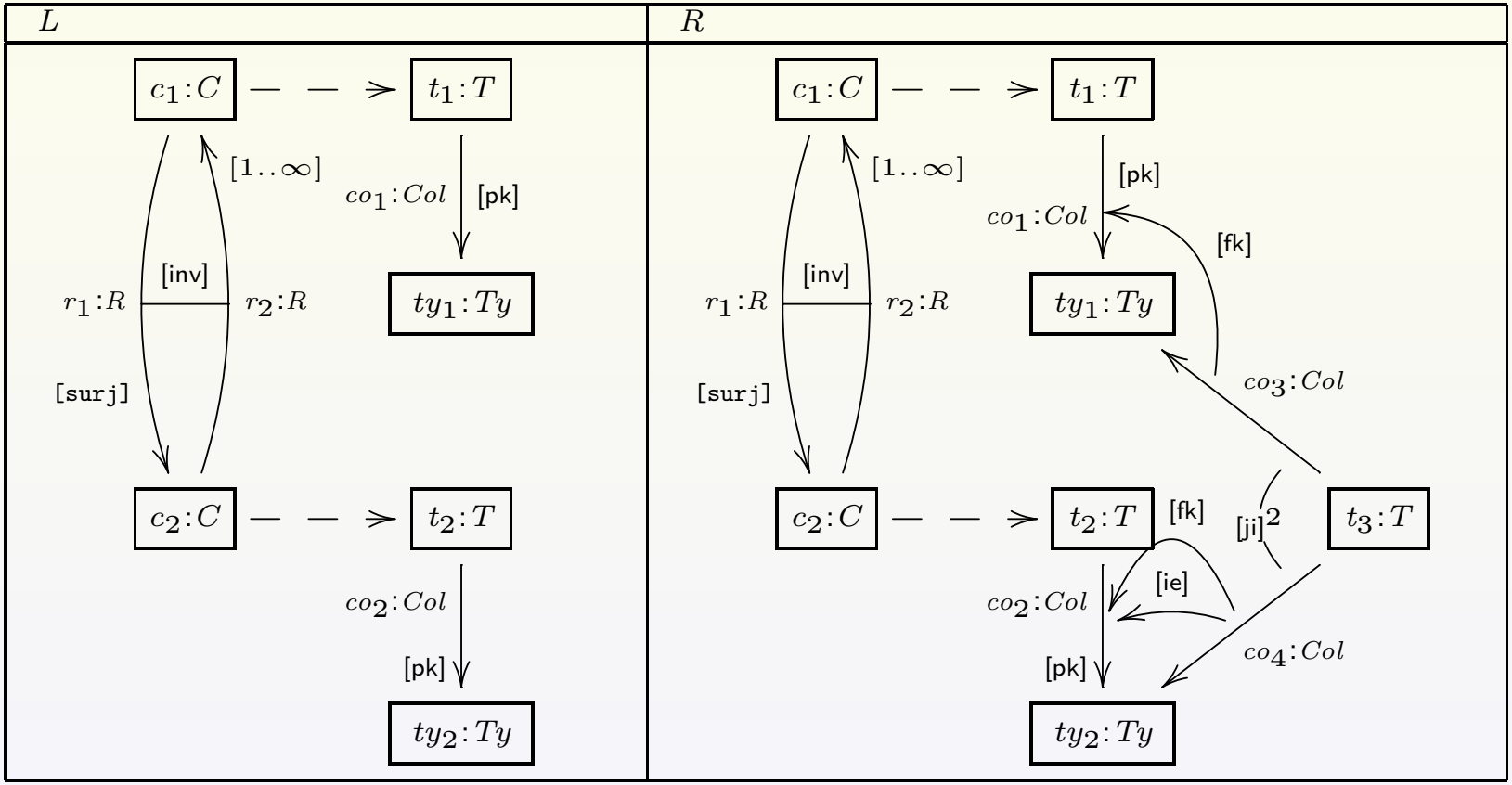
- Multiplicity one-to-many

Rule r_4: Reference to foreign key



- Multiplicity many-to-many

Rule r_5: Reference to foreign key

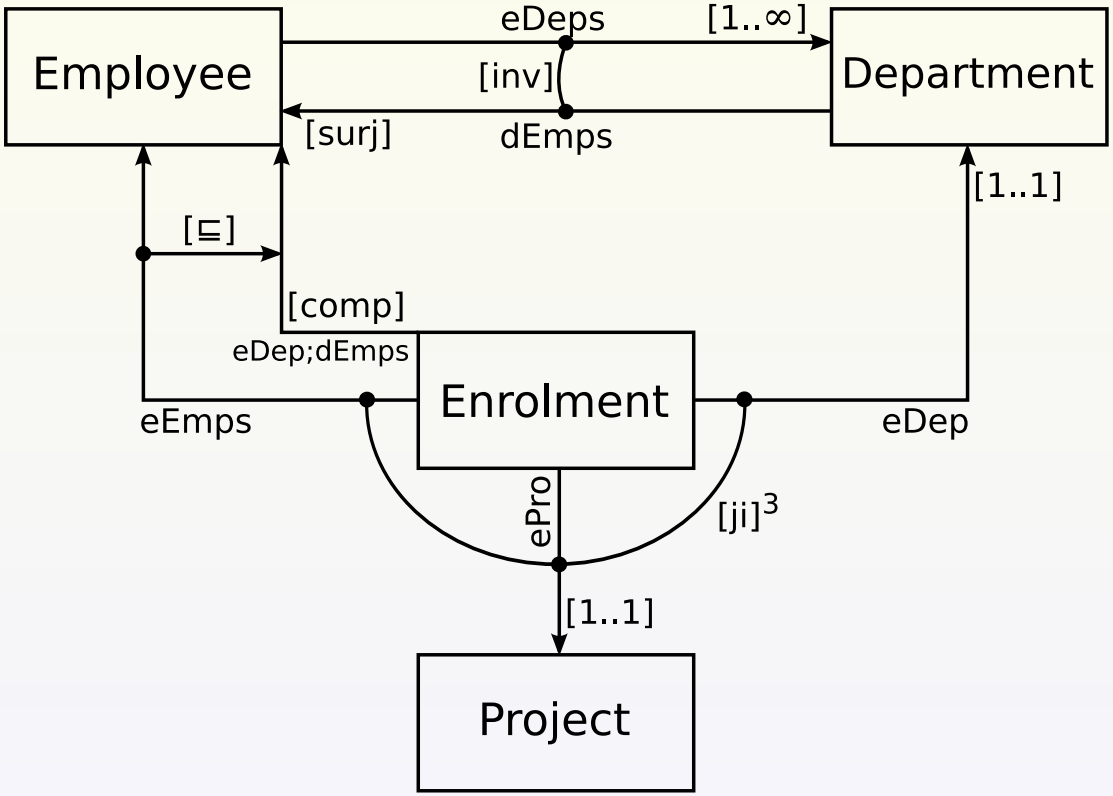


- Multiplicity many-to-many and surjective

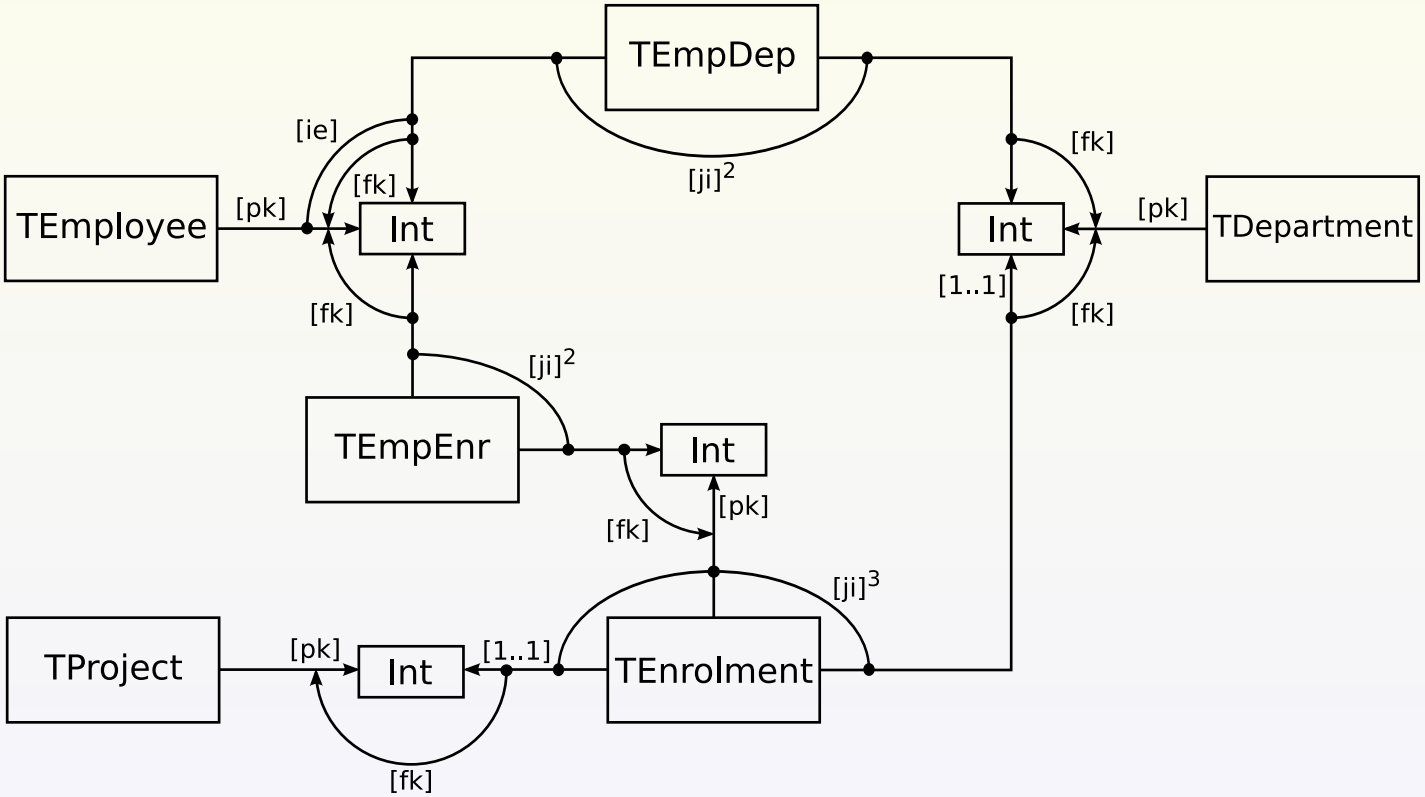
One rule application to J_1

$$
\begin{tikzcd}
 & L \arrow[r, hook, "r"] \arrow[d, "m"'] \arrow[dr, phantom, "P.O."] & R \arrow[d, "n"'] & \\
S_1 \arrow[r, hook] \arrow[dd] & J_1 \arrow[r, hook, "\langle r,m \rangle"] \arrow[ddr] & J_1' \arrow[ddl] & T_1 \arrow[l, hook'] \arrow[dd] \arrow[dl, phantom, "P.B."] \\
 & & & \\
S_2 \arrow[rr, hook] & & J_2 & T_2 \arrow[l, hook']
\end{tikzcd}
$$

Applying all the rules to this S_1...



... will lead to this T_1 after projection



Outline

Related work

- Graph Transformation System (GTS) *Hartmut Ehrig et al*
  - Adding support for transforming constraints
- Triple Graph Grammar (TGG) *Schürr et al*
  - Adding support for constraining the joined modelling formalism
- Refactoring *Fondement and Baar*
  - Transformation of constraints in refactoring UML/OCL models
  - Adding support for a more general case
- VTMS *Lengyl and Levendovzsky*
  - Validation of transformations by means of OCL constraints
  - Puts constraints on rules to narrow down matches
  - A match is a metamodel instantiation
- Analyses of GTS *Cabot et al*
  - Rules and their porperties e.g. conflicts and rule applicability to OCL expressions

Summary

- DPF: a formal diagrammatic specification framework
  - Integration of constraints in modelling
- Extending GTS and TGG: supporting transformation of attached constraints

Future work

- Controlling and scheduling transformation rules
- Dependency between predicates
- Signature morphisms: “better” joining
- Prototype tool
- Comparison to other tools: AGG, GReAT, PROGRESS, VMTS, Epsilon, Fujaba etc
- Real-size case study: noark (Norwegian Archive standard)

Thank you!

Questions?

Instances

Instances of Predicates

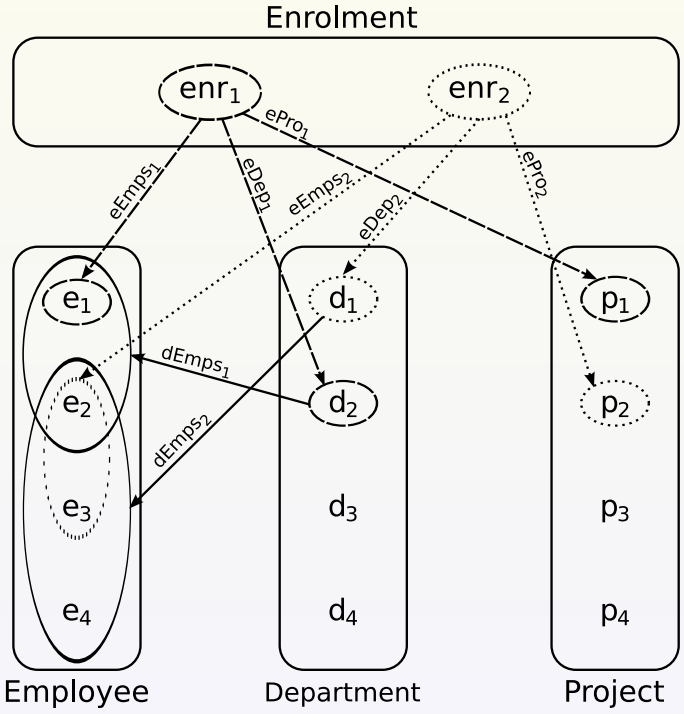
A semantic interpretation of a signature $\Sigma = (\Pi, \alpha)$ is given by a mapping that assigns to each $p \in \Pi$ a set $[\![p]\!]$ of graph homomorphisms $\tau : O \to \alpha(p)$ called valid instances of p, written $\tau \vDash p$, where O may vary over all graphs.

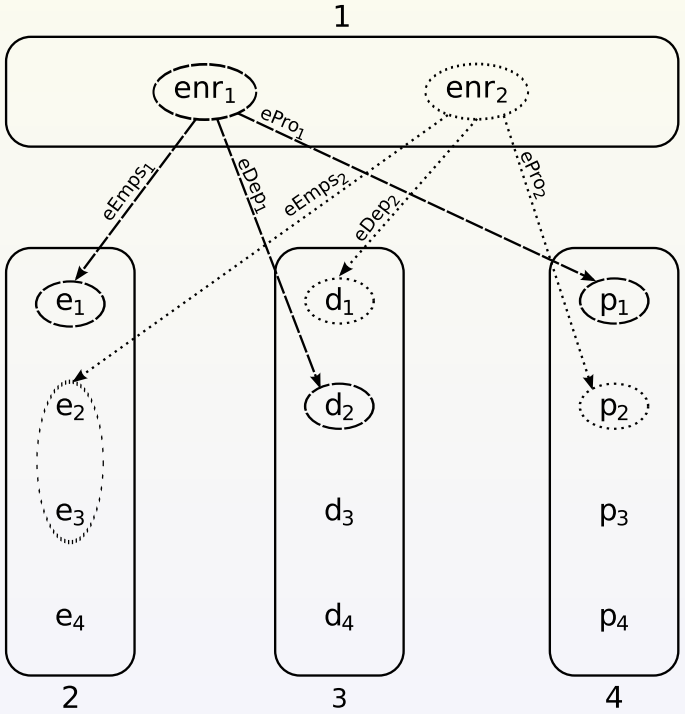
Instance of Specification

An instance of a diagrammatic specification $S = (G^S, C^S)$ is a graph I together with a graph homomorphism $\iota : I \to G^S$, written (I, ι), such that for each constraint $(p, \delta) \in C^S$ we have $\iota^* \in [\![p]\!]$, where $\iota^* : O^* \to \alpha(p)$ is given by the following pullback diagram

$$\begin{tikzcd}
\alpha(p) \arrow[r, "\delta"'] & G^S \\
O^* \arrow[u, "\iota^*"] \arrow[r, "\delta^*"'] \arrow[ur, phantom, "P.B."] & I \arrow[u, "\iota"']
\end{tikzcd}$$

Semantics of diagrammatic specification





Challenges in modelling

- Mixing graph-based structures with textual constraints
  - Different technical spaces
    - checking models in two different engines/steps
    - model-constraint synchronisation problem
    - violation of "everything-is-a-model" vision of MDE
  - Challenge for domain experts who do not understand OCL

Formalisation approach

- Based on category theory
  - Sketches formalism: define semantics of diagrams (thus graph-based models)
    - models: graphs (nodes and edges)
    - model properties: universal properties (limit, colimit, commutative diagrams)
  - Generalized sketches formalism
    - not only universal properties
    - user-defined diagrammatic predicate signatures
- DPF: specification formalism based on generalized sketches