Formal and Relational Concept Analysis approaches in Software Engineering: an overview and an application to learn model transformation patterns in examples

Xavier Dolques, Marianne Huchard, Clémentine Nebut, Hajer Saada

Author manuscript, published in "ICESE'11: First ICESE Virtual Workshop, Search-based Model-Driven Engineering, Qatar (2011)"

Formal and Relational Concept Analysis approaches in Software Engineering:

an overview and an application to learn model transformation patterns in examples

Xavier Dolques, Marianne Huchard,
Clémentine Nebut, Hajer Saada

[1] INRIA, Centre Inria Rennes - Bretagne Atlantique
[2] LIRMM - Université Montpellier 2, CNRS

First ICESE Virtual Workshop on Software Engineering and Artificial Intelligence, may 2011

Outline

Motivations

Software contains plenty of data analysis problems involved in

- the forward engineering process
- the re-engineering tasks
- various analyses

Focusing on Formal Concept Analysis

- an exploratory data analysis / data mining method
- an unsupervised machine learning approach
- produces clusters, classification and implication rules

Motivations

Highlight main characteristics of FCA

- defining FCA
- main applications of FCA in SE
- multi-relational data analysis with RCA
- young applications of RCA in SE

Learning model transformation (MT) patterns

- on examples of MT
- building of MT examples using ontology alignment
- learning MT patterns with RCA

Outline

Formal Concept Analysis (FCA)

What is FCA?

- a formalization of the philosophical notion of *concept*
- an approach for data analysis and knowledge processing
- many existing experiences and projects
- algorithms, graphical representations, tools
- an active research community (3 conf. ICFCA, CLA, ICCS)

source http://people.aifb.kit.edu/jvo/fca4sw/

Formal Concept Analysis (FCA)

What is a concept?

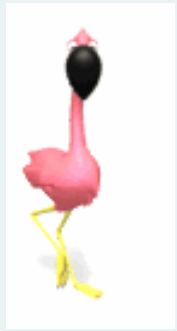
The concept *bird*

- a set of objects (concept's extent):

- a set of attributes / characteristics (concept's intent):
  feathers, with a bill, etc.

Formal Concept Analysis (FCA)

How concepts are organized?



The concept *flamingo* is a subconcept of the concept *bird*

- inclusion of concept's extents:
  the set of flamingos is included in the set of birds
- inclusion of the concept's intents:
  the attributes of birds are included in the attributes of flamingos

Formal Concept Analysis (FCA)

The formal context

Things that are known about the world

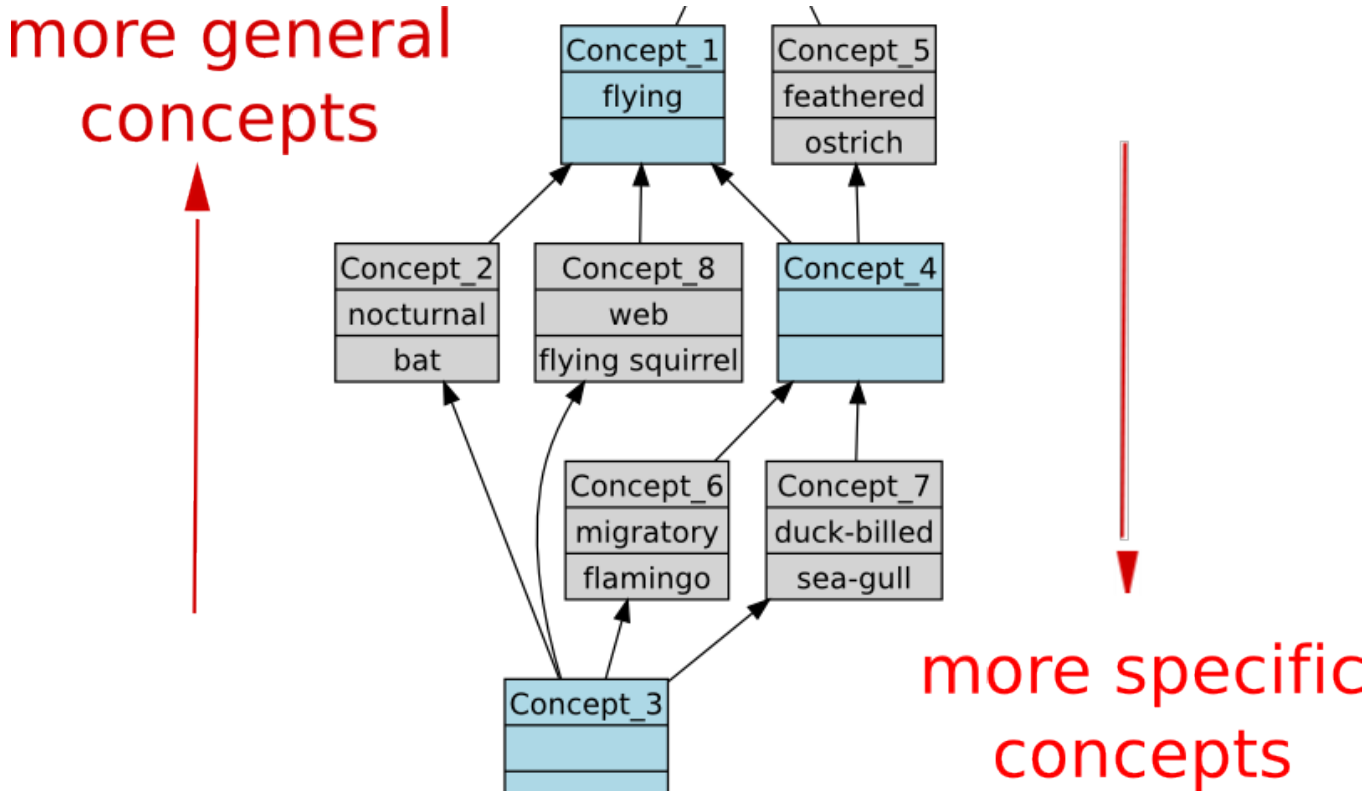
	flying (fl)	nocturnal (n)	feathered (fe)	migratory (m)	duck-billed (db)	web (w)
flying squirrel (S)	x					x
bat (B)	x	x				
ostrich (O)			x			
flamingo (F)	x		x	x		
sea-gull (G)	x		x		x	

Concepts can be derived from a formal context

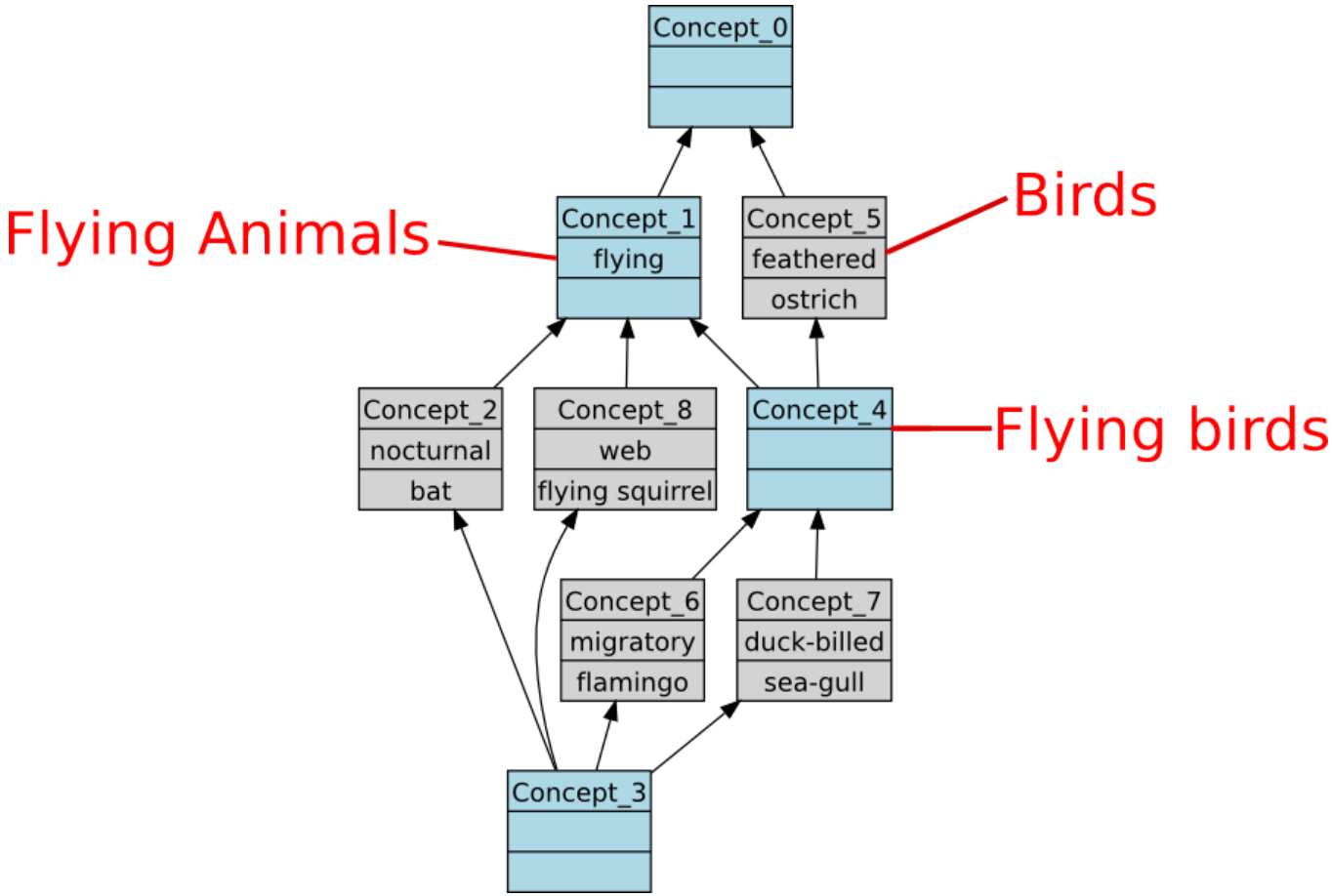
A formal concept is a pair (X, Y) where

- Y is the set of attributes common to the objects of X
- X is the set of objects having all attributes of Y

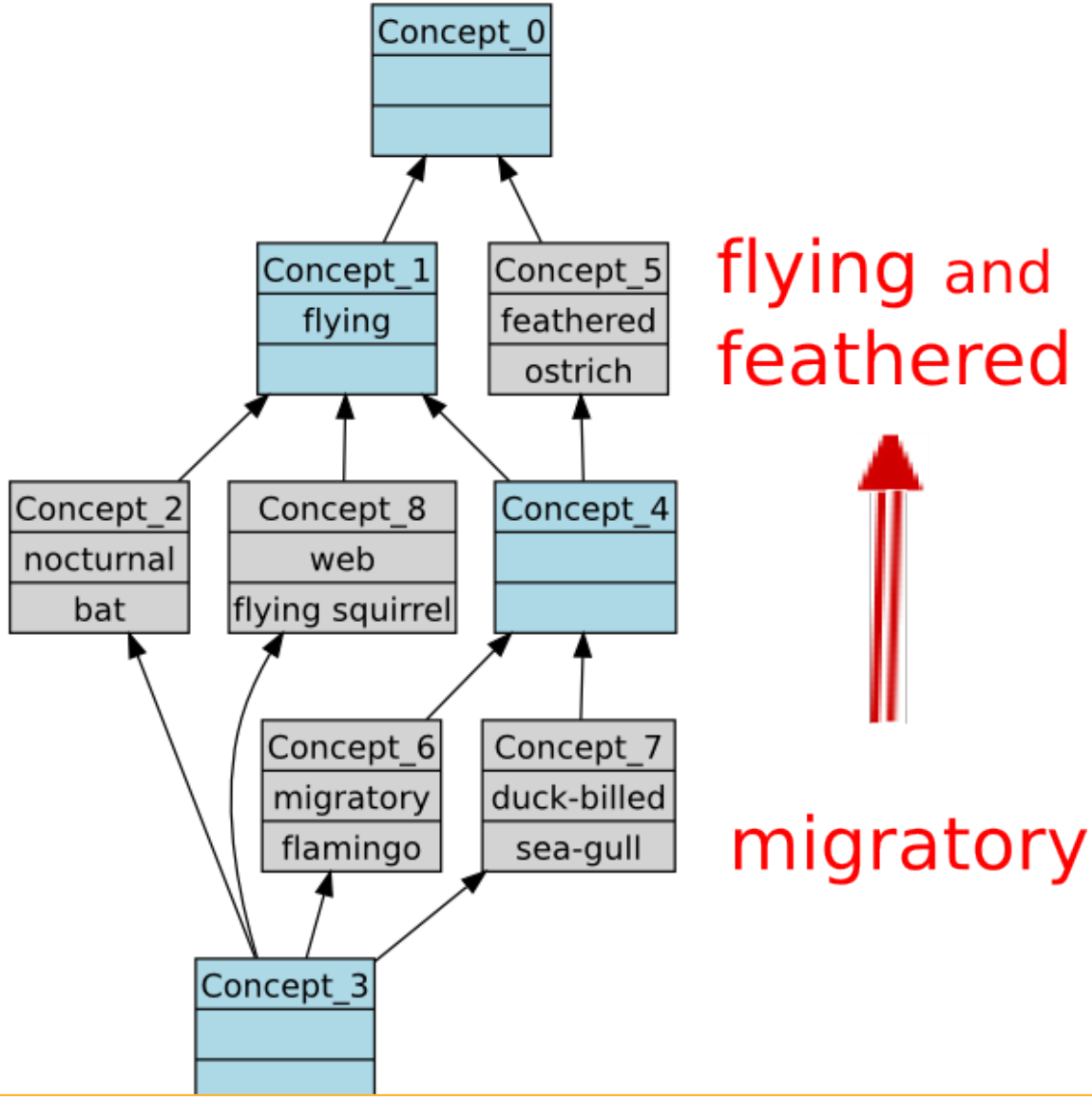
The concept lattice: specialization order



The concept lattice: clusters



The concept lattice: implication rules



FCA in Software engineering

A Survey of Formal Concept Analysis Support for Software Engineering Activities, Tilley et al., FCA 2005

... And many research work during the past 5 years

- Requirement Analysis: elaborating requirements [Andelfinger], reconciling stake-holders [Düwel et al.], linking use cases and classes [Böttger et al.]
- Component / Web service classification and retrieval [Lindig, Fisher, Aboud et al., Azmeh et al.]
- Exploring a formal specification [Tilley]
- Dynamic analysis: debug temporal specifications [Ammons et al.], test coverage [Ball], locating features [Eisenbarth et al., Bojic et al.], fault localization [Cellier et al.]

FCA in Software engineering

- Analysis of legacy systems:
  - Configuration structure [Snelting]
  - Grouping fields in COBOL systems [Van Deursen et al., Kuipers et al.]
  - Migrating COBOL towards Corba components [Canfora]
  - Migrating from imperative to OO paradigm [Sahraoui et al., Siff et al., Tonella]
  - Reengineering class hierarchies [Snelting et al., Schupp et al., Godin et al., Huchard et al.]
  - Detecting patterns [Tonella and Antoniol, Arévalo et al.]
  - Order for reading classes [Dekel]
  - Bad smell correction [Bhatti et al.]
  - Conceptual code exploration [Cole et al.]
  - Aspect mining [Tonella and Ceccato, Tourwé and Mens]
  - Access-guided client class extraction for Eiffel [Ardourel and Huchard]
  - Mining Source Code for Structural Regularities [Lozano et al.]

Outline

Relational Concept Analysis (RCA)

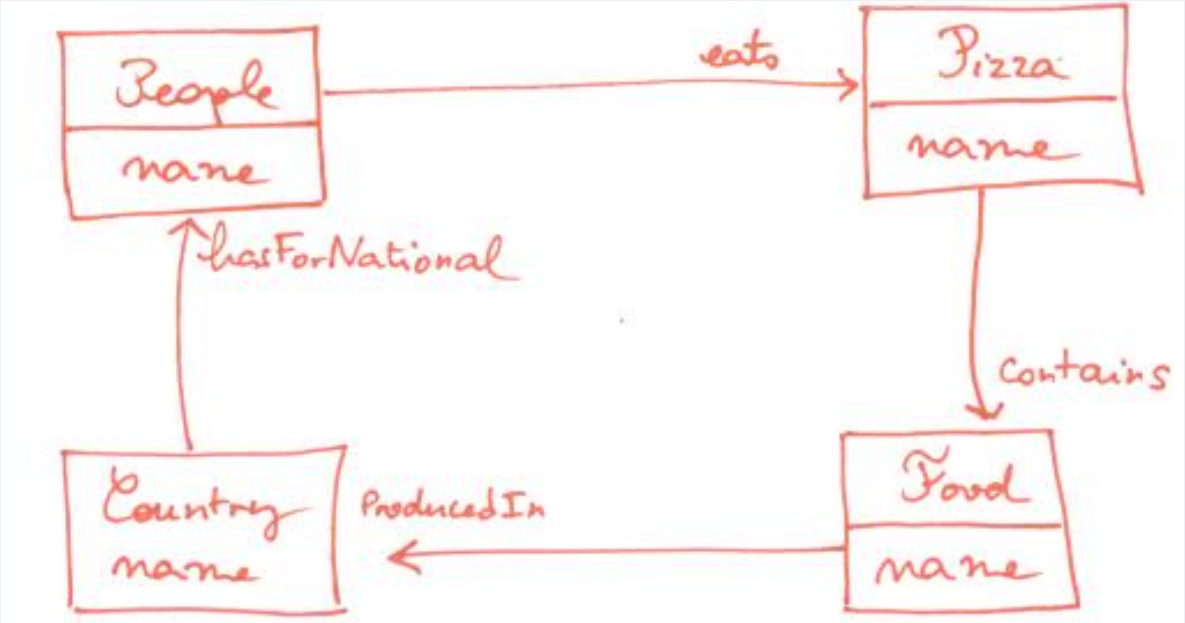
- Extend the purpose of FCA for taking into account relations between objects
- The RCA process relies on the following main points:
  - a relational model based on the entity-relationship model
  - a conceptual scaling process allowing to represent relations between objects as relational attributes
  - an iterative process for designing a concept lattice where concept intents include non-relational and relational attributes.
- RCA provides relational structures that can be represented as ontology concepts within a knowledge representation formalism such as description logics (DLs).

Huchard, M., Hacene M. R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. 49(1-4): 39-76 (2007)

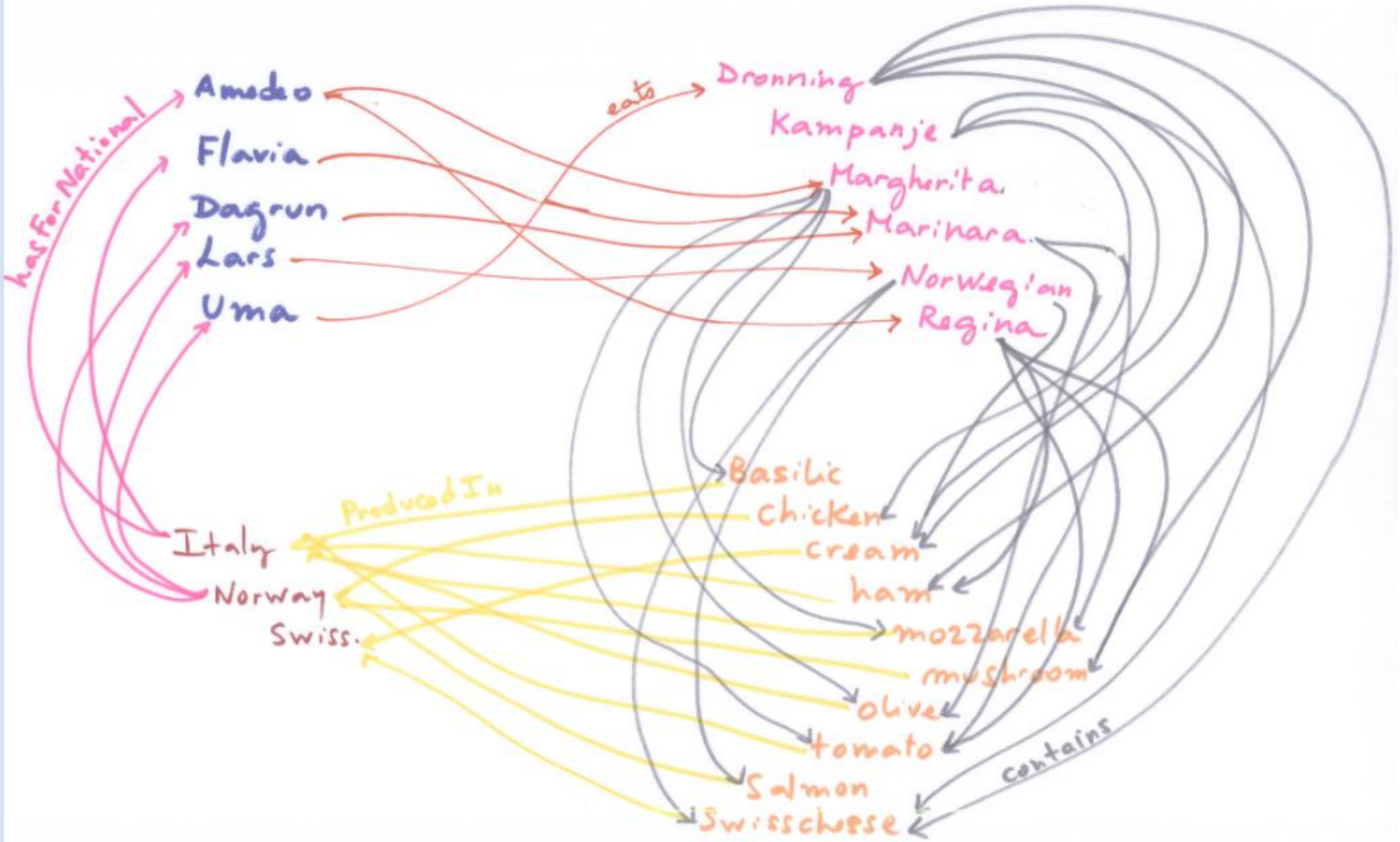
Relational Concept Analysis (RCA)

A relational model based on the entity-relationship model ...

Pizza story

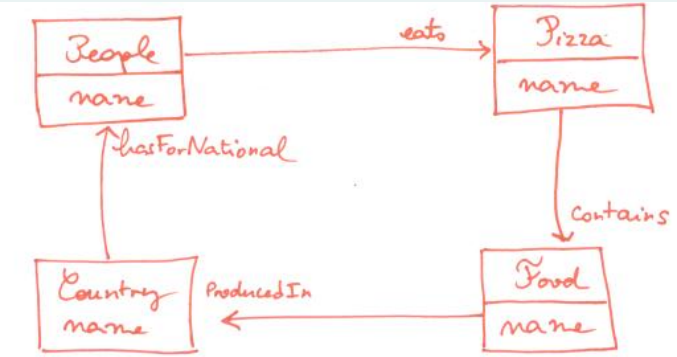


Objects and links



Relational Concept Analysis (RCA)

Pizza story



Pizza data

- four object/attribute contexts
  - $K_{People} \subset$ People $\times$ people names
  - $K_{Pizza} \subset$ Pizza $\times$ pizza names
  - $K_{Food} \subset$ Food item $\times$ food names
  - $K_{Country} \subset$ Country $\times$ country names
- four object/object contexts
  - `eats` $\subset$ People $\times$ Pizza
  - `contains` $\subset$ Pizzas $\times$ Food item
  - `producedIn` $\subset$ Food item $\times$ Country
  - `hasForNational` $\subset$ Country $\times$ People

Formal contexts

K_{People}	Amedeo	Flavia	Dagrun	Lars	Uma
Amedeo	×				
Flavia		×			
Dagrun			×		
Lars				×	
Uma					×

K_{Pizza}	Dronning	Kampanje	Margherita	Marina	Norwegian	Regina
Dronning	×					
Kampanje		×				
Margherita			×			
Marina				×		
Norwegian					×	
Regina						×

$K_{Ingredients}$	basilic	chicken	cream	ham	mozzarella	mushroom	olive	tomato	salmon	swisscheese
basilic	×									
chicken		×								
cream			×							
ham				×						
mozzarella					×					
mushroom						×				
olive							×			
tomato								×		
salmon									×	
swisscheese										×

$K_{Country}$	Italy	Norway	Switzerland
Italy	×		
Norway		×	
Switzerland			×

Relational context: R_{eats}

	Dronning	Kampanje	Margherita	Marina	Norwegian	Regina
Amedeo			×			×
Flavia				×		
Dagrun				×		×
Lars					×	
Uma	×				×	

Relational context: $R_{contains}$

	basilic	chicken	cream	ham	mozzarella	mushroom	olive	tomato	salmon	swisscheese
Dronning			×	×		×				×
Kampanje		×	×							×
Margherita	×				×		×	×		
Marina							×	×		
Norwegian			×						×	×
Regina				×	×	×		×		

Relational context: $R_{producedIn}$

	Italy	Norway	Switzerland
basilic	×		
chicken		×	
cream			×
ham	×		
mozzarella	×		
mushroom		×	
olive	×		
tomato	×		
salmon		×	
swisscheese			×

Relational context: $R_{hasForNational}$

	Amedeo	Flavia	Dagrun	Lars	Uma
Italy	×	×			
Norway			×	×	×
Switzerland					

Relational Context Family (RCF)

A RCF $\mathcal{F}$ is a pair (K, R) with:

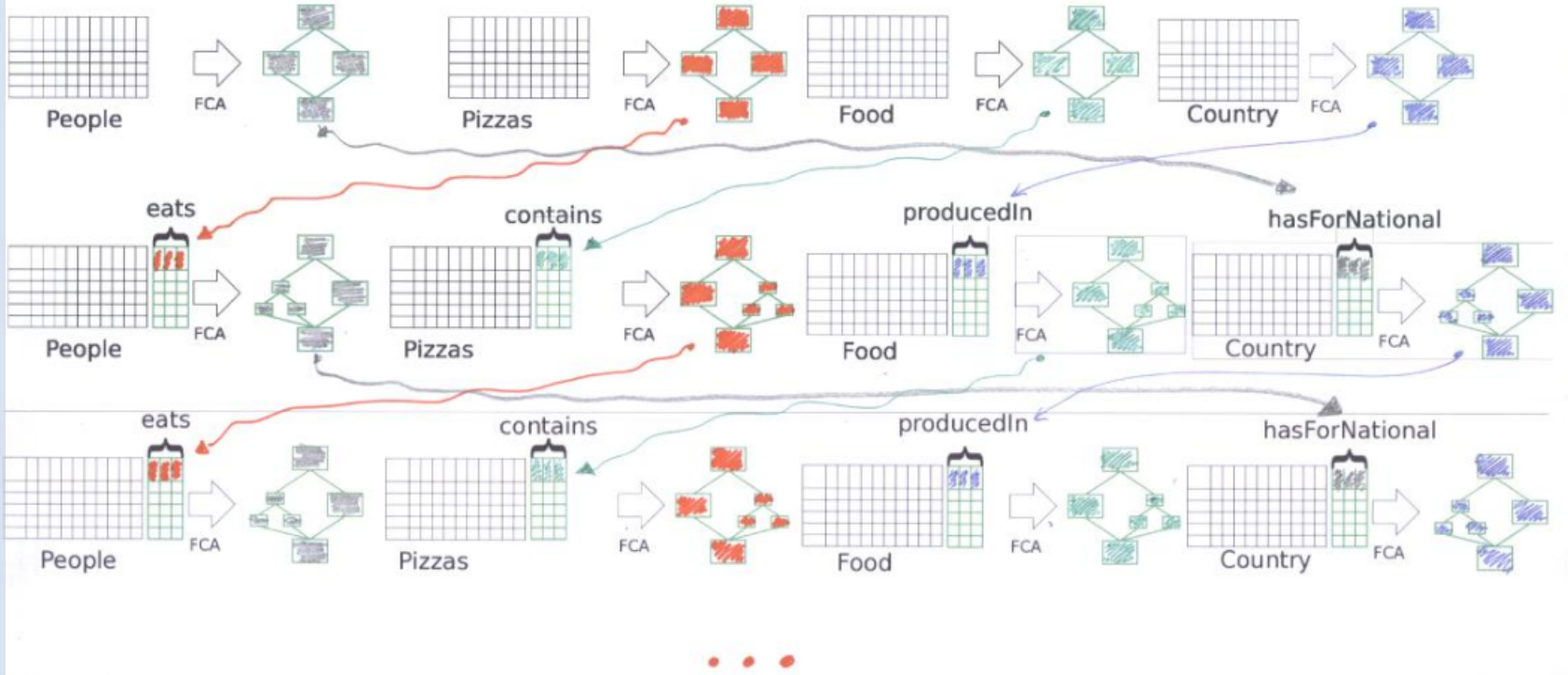
- K is a set of formal contexts $K_i = (O_i, A_i, I_i)$
- R is a set of relational contexts $R_j = (O_k, O_l, I_j)$,

Pizza RCF

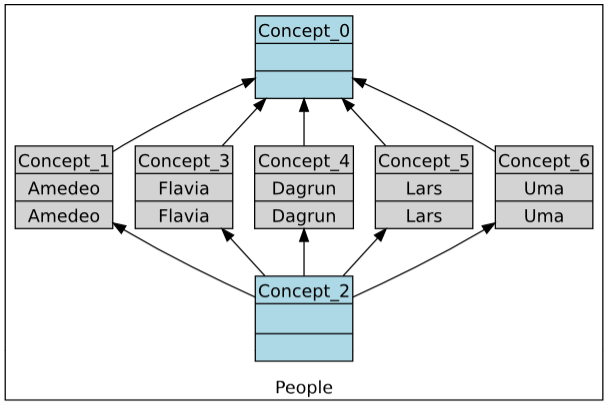
$K = K_{People}, K_{Pizza}, K_{Food}, K_{Country}$
$R = R_{eats}, R_{contains}, R_{producedIn}, R_{hasForNational}$

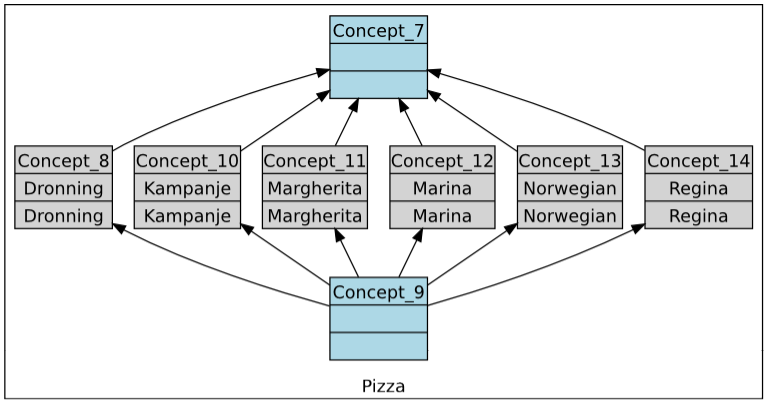
An iterative approach (RCA)

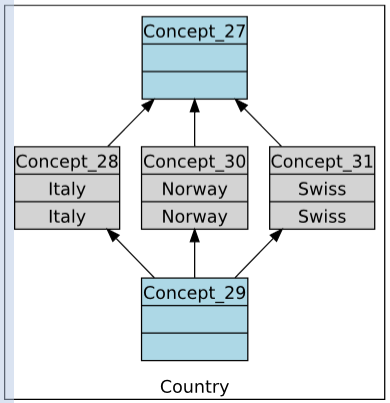
Learned concepts are used in a next step to learn more

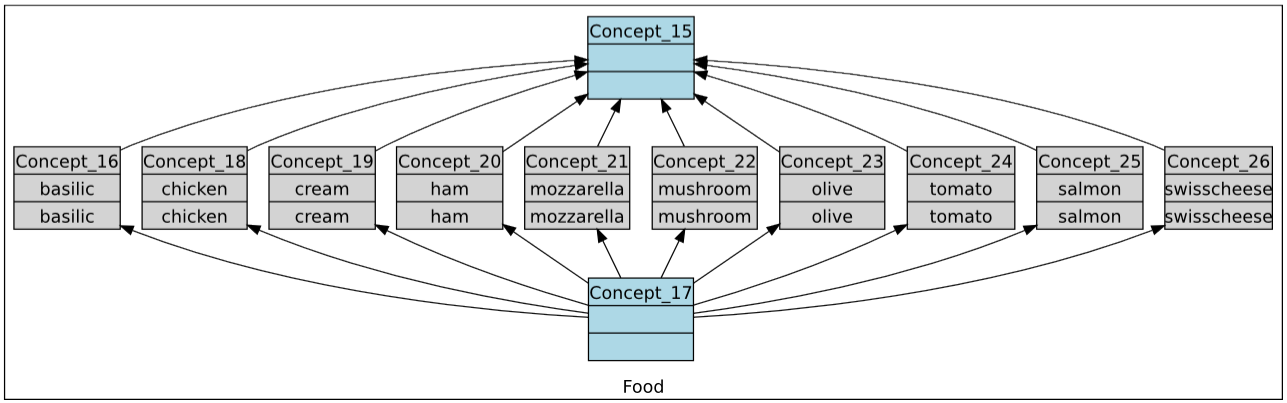


RCA - Step 0 - Initial Lattices









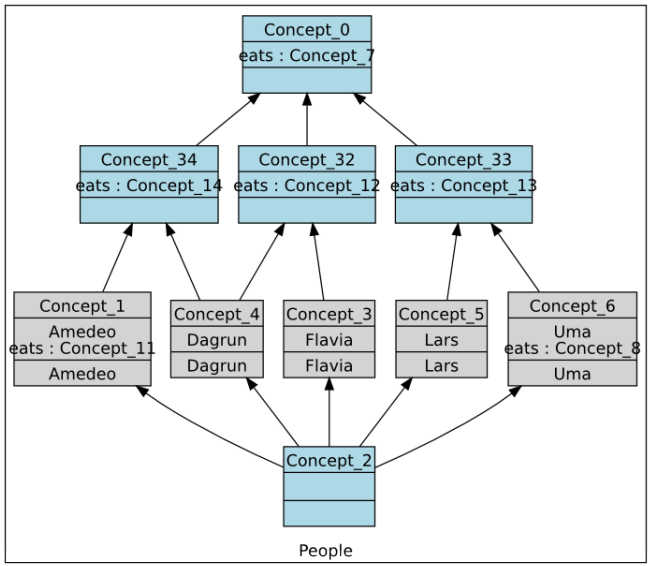
Scaling relations

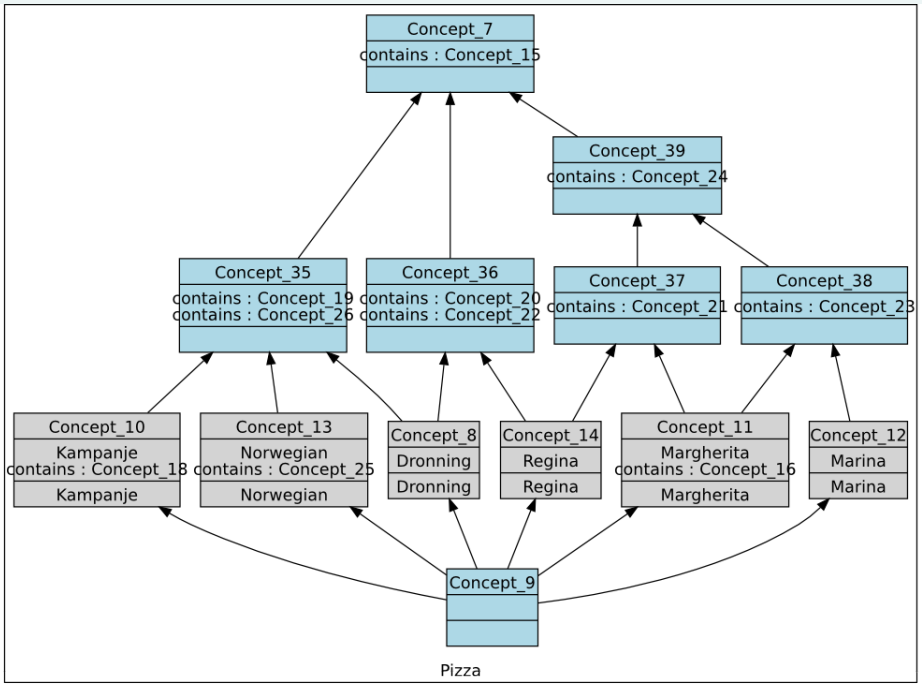
Integrating concepts in the relational contexts

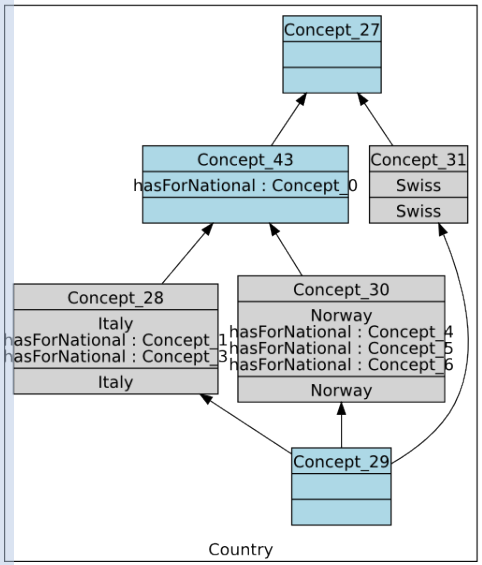
Amedeo eats Margherita ; Margherita $\in$ extent(Concept_11)
$\rightarrow \exists p \in$Concept_11, s.t. Amedeo eats p
$\rightarrow$ (Amedeo, eats:Concept_11)
$\rightarrow$ (Amedeo, Concept_11) belongs to the existentially scaled relation eat*, (Amedeo,$\exists$ *eat* :Concept_11) stands

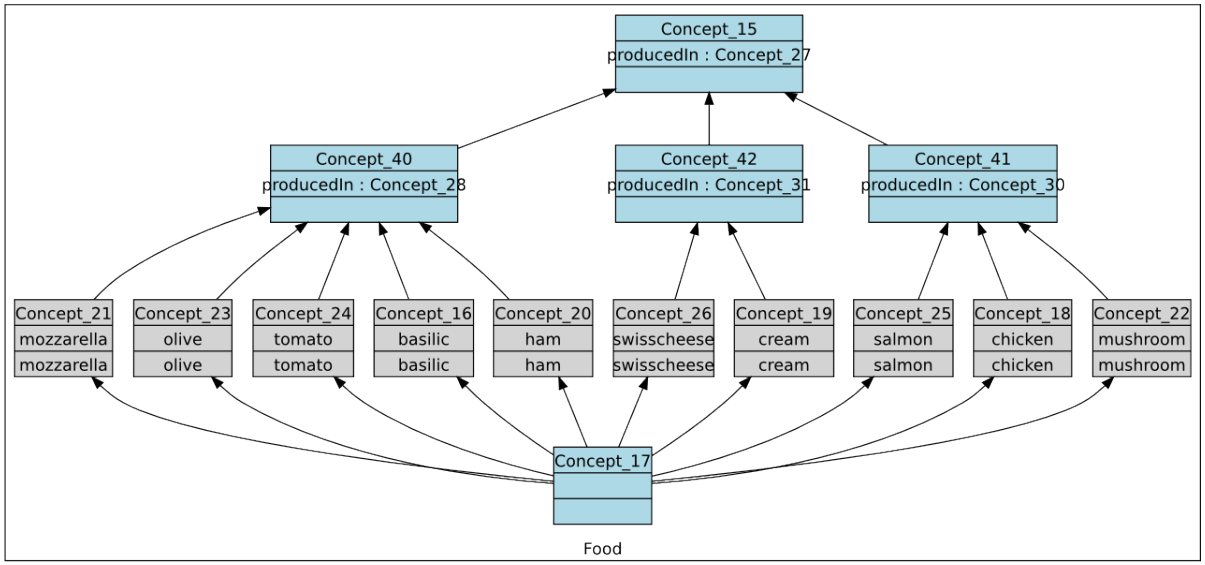
	Amedeo	Flavia	Dagrun	Lars	Uma		eats : Concept_7	eats : Concept_8	eats : Concept_11	eats : Concept_12	eats : Concept_13	eats : Concept_14
Amedeo	X						X		X			X
Flavia		X					X			X		
Dagrun			X				X			X		X
Lars				X			X				X	
Uma					X		X	X			X	

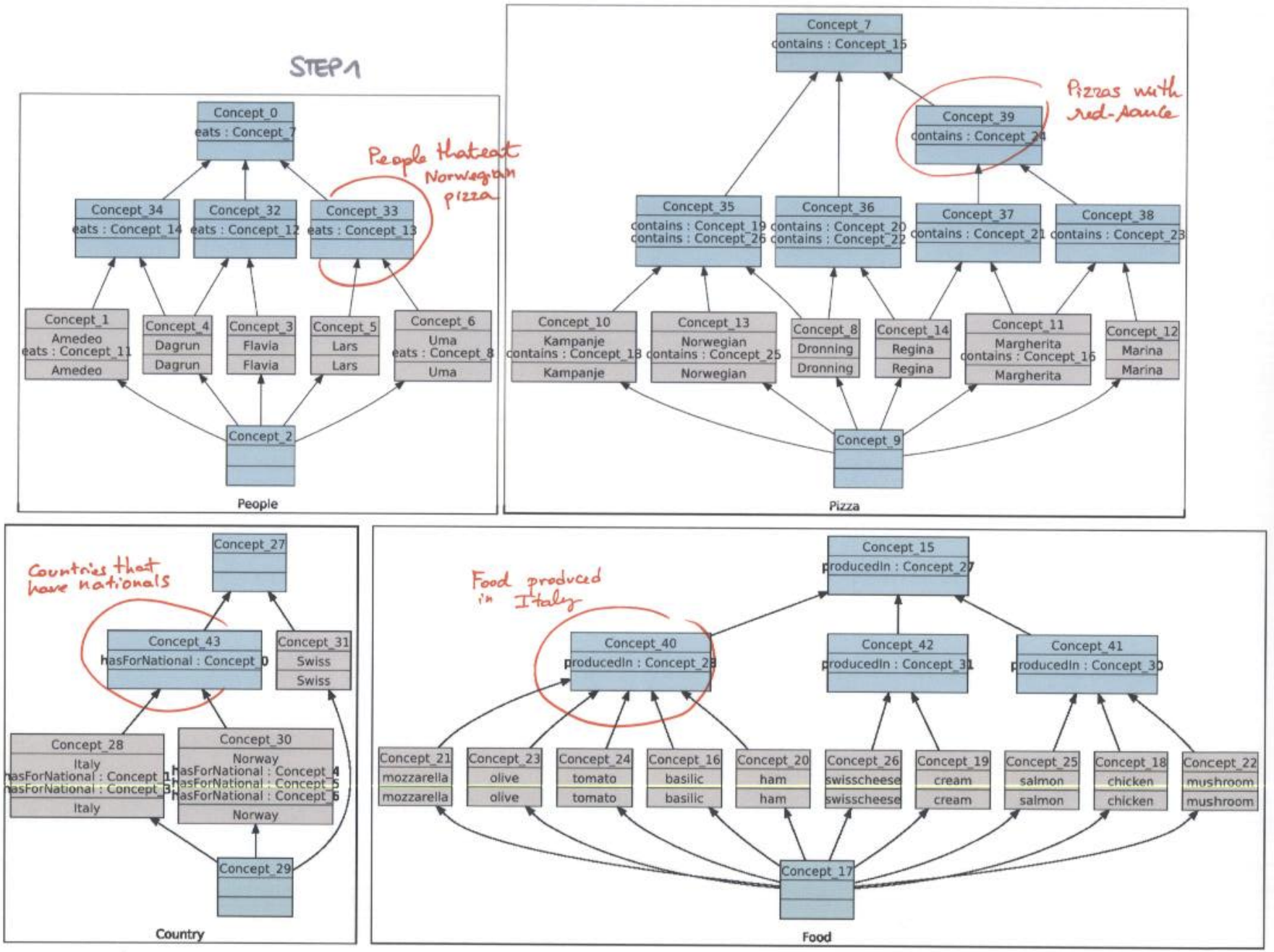
RCA - Lattices at step 1



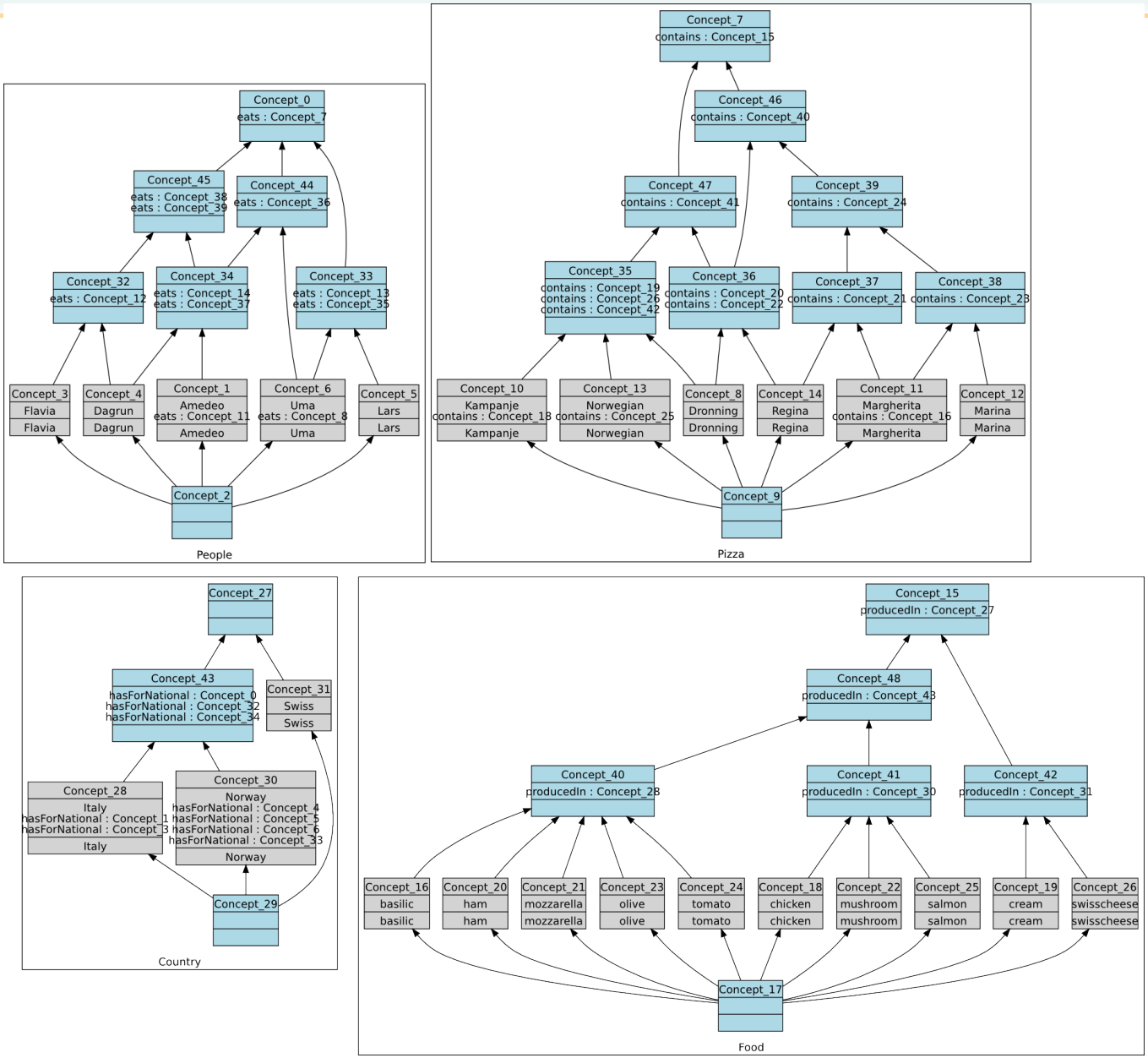


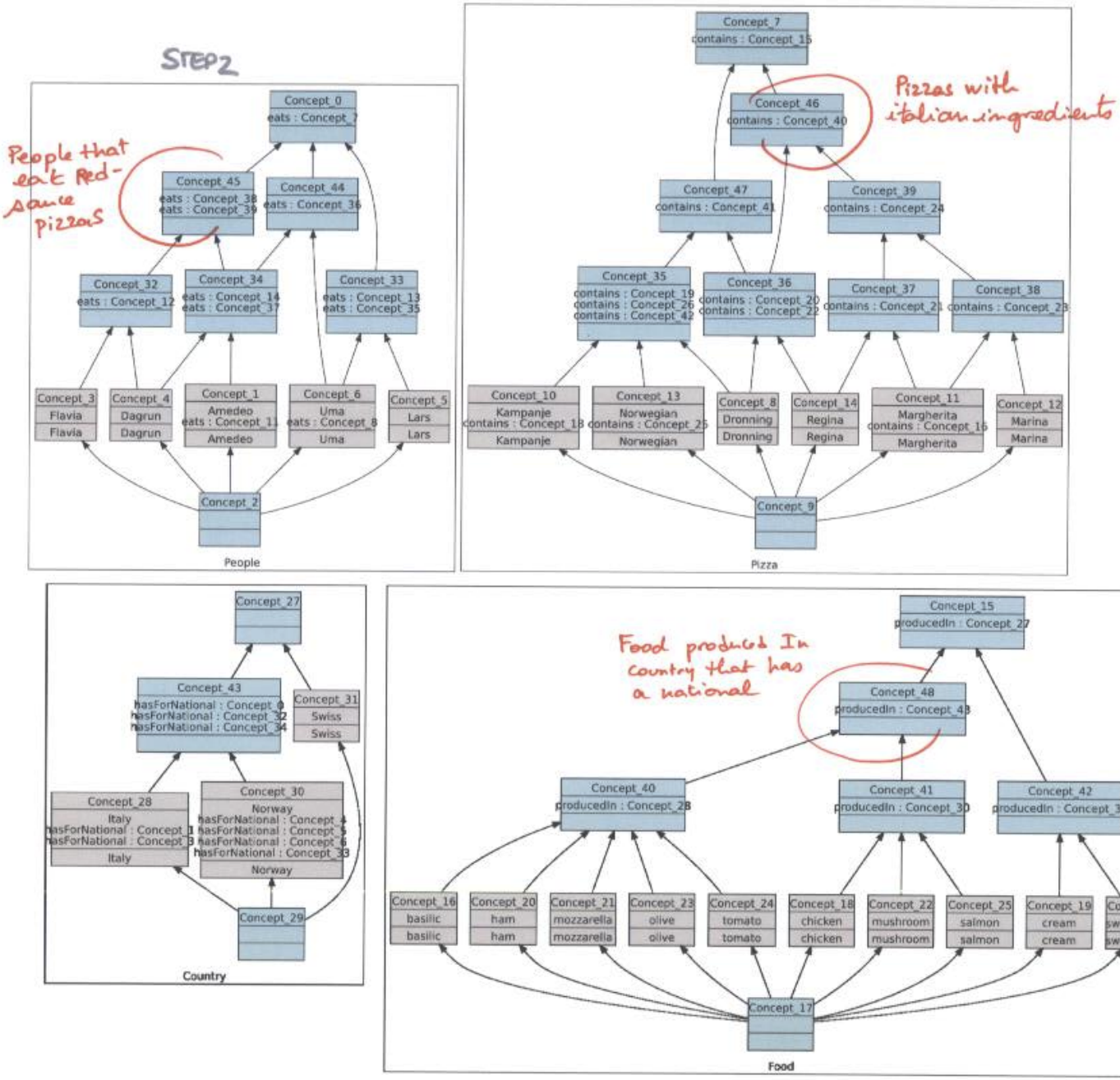




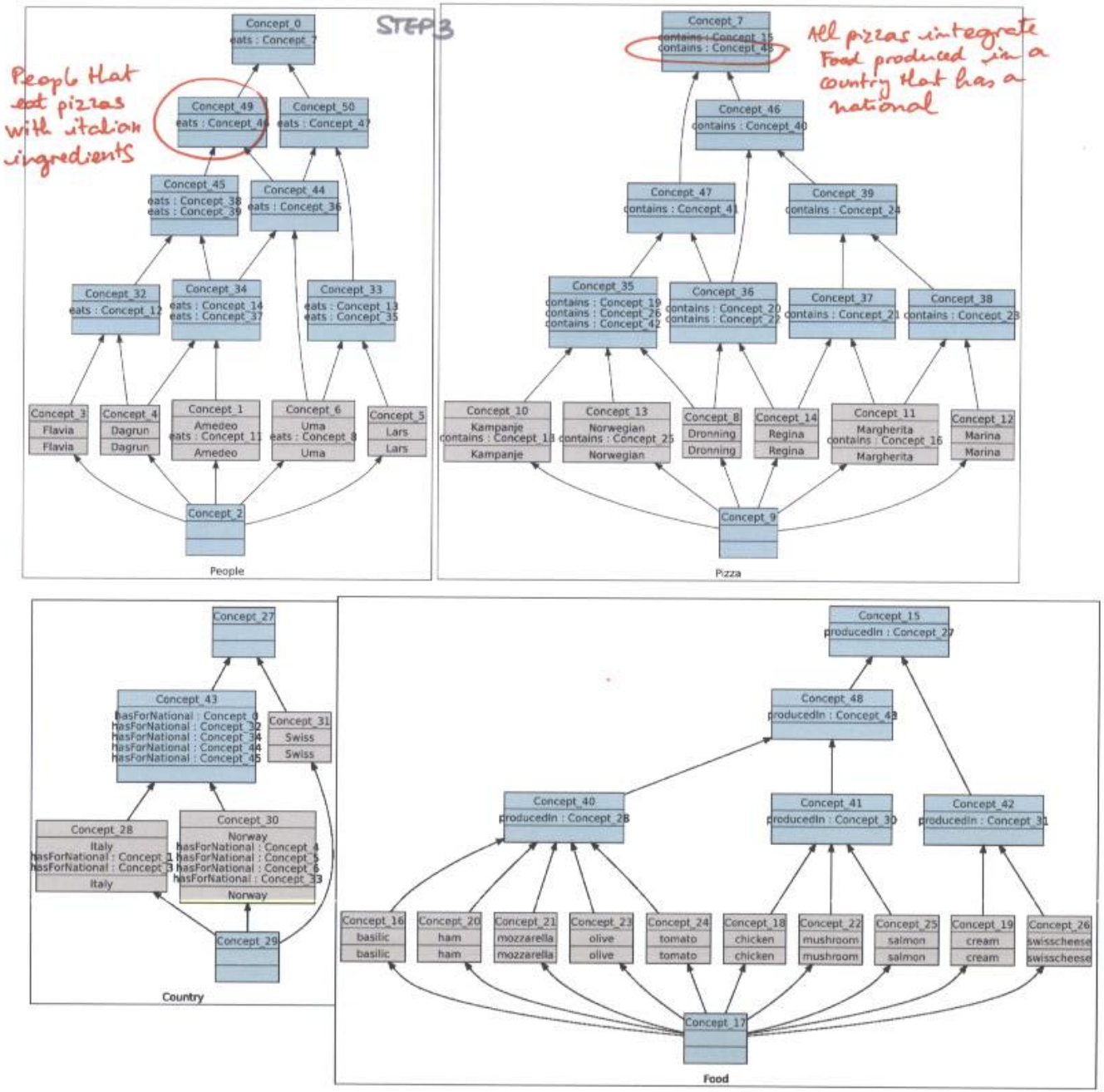

STEP 1
Concept_0
eats : Concept_7
People that eat Norwegian pizza
Concept_34
eats : Concept_14
Concept_32
eats : Concept_12
Concept_33
eats : Concept_13
Concept_1
Amedeo
eats : Concept_11
Amedeo
Concept_4
Dagrun
Dagrun
Concept_3
Flavia
Flavia
Concept_5
Lars
Lars
Concept_6
Uma
eats : Concept_8
Uma
Concept_2
People
Concept_7
contains : Concept_15
Pizzas with red-sauce
Concept_39
contains : Concept_24
Concept_35
contains : Concept_19
contains : Concept_26
Concept_36
contains : Concept_20
contains : Concept_22
Concept_37
contains : Concept_21
Concept_38
contains : Concept_23
Concept_10
Kampanje
contains : Concept_18
Kampanje
Concept_13
Norwegian
contains : Concept_25
Norwegian
Concept_8
Dronning
Dronning
Concept_14
Regina
Regina
Concept_11
Margherita
contains : Concept_16
Margherita
Concept_12
Marina
Marina
Concept_9
Pizza
Countries that have nationals
Concept_27
Concept_43
hasForNational : Concept_0
Concept_31
Swiss
Swiss
Concept_28
Italy
hasForNational : Concept_1
hasForNational : Concept_3
Italy
Concept_30
Norway
hasForNational : Concept_4
hasForNational : Concept_5
hasForNational : Concept_6
Norway
Concept_29
Country
Food produced in Italy
Concept_15
producedIn : Concept_27
Concept_40
producedIn : Concept_28
Concept_42
producedIn : Concept_31
Concept_41
producedIn : Concept_30
Concept_21
mozzarella
mozzarella
Concept_23
olive
olive
Concept_24
tomato
tomato
Concept_16
basilic
basilic
Concept_20
ham
ham
Concept_26
swisscheese
swisscheese
Concept_19
cream
cream
Concept_25
salmon
salmon
Concept_18
chicken
chicken
Concept_22
mushroom
mushroom
Concept_17
Food

RCA - Lattices at step 2

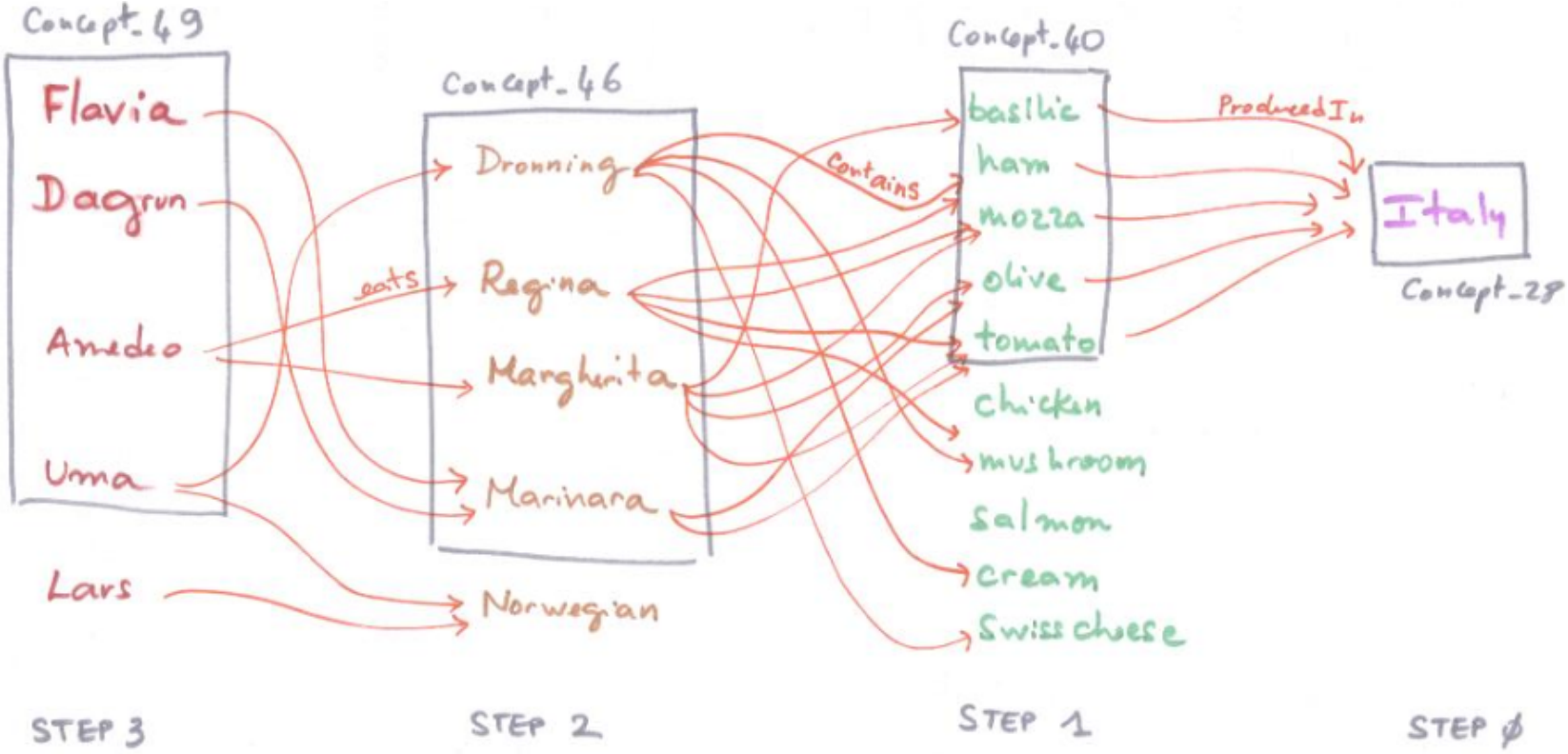



STEP2
People that eat Red-Sauce pizzas
Concept_0
eats : Concept_7
Concept_45
eats : Concept_38
eats : Concept_39
Concept_44
eats : Concept_36
Concept_32
eats : Concept_12
Concept_34
eats : Concept_14
eats : Concept_37
Concept_33
eats : Concept_13
eats : Concept_35
Concept_3
Flavia
Flavia
Concept_4
Dagrun
Dagrun
Concept_1
Amedeo
eats : Concept_11
Amedeo
Concept_6
Uma
eats : Concept_8
Uma
Concept_5
Lars
Lars
Concept_2
People
Concept_7
contains : Concept_15
Concept_46
contains : Concept_40
Pizzas with italian ingredients
Concept_47
contains : Concept_41
Concept_39
contains : Concept_24
Concept_35
contains : Concept_19
contains : Concept_26
contains : Concept_42
Concept_36
contains : Concept_20
contains : Concept_22
Concept_37
contains : Concept_21
Concept_38
contains : Concept_23
Concept_10
Kampanje
contains : Concept_18
Kampanje
Concept_13
Norwegian
contains : Concept_25
Norwegian
Concept_8
Dronning
Dronning
Concept_14
Regina
Regina
Concept_11
Margherita
contains : Concept_16
Margherita
Concept_12
Marina
Marina
Concept_9
Pizza
Concept_27
Concept_43
hasForNational : Concept_0
hasForNational : Concept_32
hasForNational : Concept_34
Concept_31
Swiss
Swiss
Concept_28
Italy
hasForNational : Concept_1
hasForNational : Concept_3
Italy
Concept_30
Norway
hasForNational : Concept_4
hasForNational : Concept_5
hasForNational : Concept_6
hasForNational : Concept_33
Norway
Concept_29
Country
Food produced In country that has a national
Concept_15
producedIn : Concept_27
Concept_48
producedIn : Concept_43
Concept_40
producedIn : Concept_28
Concept_41
producedIn : Concept_30
Concept_42
producedIn : Concept_31
Concept_16
basilic
basilic
Concept_20
ham
ham
Concept_21
mozzarella
mozzarella
Concept_23
olive
olive
Concept_24
tomato
tomato
Concept_18
chicken
chicken
Concept_22
mushroom
mushroom
Concept_25
salmon
salmon
Concept_19
cream
cream
Concept_26
swisscheese
swisscheese
Concept_17
Food

RCA - Lattices at step 3


STEP3
People that eat pizzas with italian ingredients
All pizzas integrate Food produced in a country that has a national
Concept_0
eats : Concept_7
Concept_49
eats : Concept_46
Concept_50
eats : Concept_47
Concept_45
eats : Concept_38
eats : Concept_39
Concept_44
eats : Concept_36
Concept_32
eats : Concept_12
Concept_34
eats : Concept_14
eats : Concept_37
Concept_33
eats : Concept_13
eats : Concept_35
Concept_3
Flavia
Flavia
Concept_4
Dagrun
Dagrun
Concept_1
Amedeo
eats : Concept_11
Amedeo
Concept_6
Uma
eats : Concept_8
Uma
Concept_5
Lars
Lars
Concept_2
People
Concept_7
contains : Concept_15
contains : Concept_48
Concept_46
contains : Concept_40
Concept_47
contains : Concept_41
Concept_39
contains : Concept_24
Concept_35
contains : Concept_19
contains : Concept_26
contains : Concept_42
Concept_36
contains : Concept_20
contains : Concept_22
Concept_37
contains : Concept_21
Concept_38
contains : Concept_23
Concept_10
Kampanje
contains : Concept_18
Kampanje
Concept_13
Norwegian
contains : Concept_25
Norwegian
Concept_8
Dronning
Dronning
Concept_14
Regina
Regina
Concept_11
Margherita
contains : Concept_16
Margherita
Concept_12
Marina
Marina
Concept_9
Pizza
Concept_27
Concept_43
hasForNational : Concept_0
hasForNational : Concept_32
hasForNational : Concept_34
hasForNational : Concept_44
hasForNational : Concept_45
Concept_31
Swiss
Swiss
Concept_28
Italy
hasForNational : Concept_1
hasForNational : Concept_3
Italy
Concept_30
Norway
hasForNational : Concept_4
hasForNational : Concept_5
hasForNational : Concept_6
hasForNational : Concept_33
Norway
Concept_29
Country
Concept_15
producedIn : Concept_27
Concept_48
producedIn : Concept_43
Concept_40
producedIn : Concept_28
Concept_41
producedIn : Concept_30
Concept_42
producedIn : Concept_31
Concept_16
basilic
basilic
Concept_20
ham
ham
Concept_21
mozzarella
mozzarella
Concept_23
olive
olive
Concept_24
tomato
tomato
Concept_18
chicken
chicken
Concept_22
mushroom
mushroom
Concept_25
salmon
salmon
Concept_19
cream
cream
Concept_26
swisscheese
swisscheese
Concept_17
Food

An excerpt of the iteration



Applications

- UML class diagram refactoring

  * M. Dao, M. Huchard, M. Rouane-Hacene, C. Roume, P. Valtchev: Improving Generalization Level in UML Models Iterative Cross Generalization in Practice. ICCS 2004: 346-360

  * G. Arévalo, J.-R. Falleri, M. Huchard, C. Nebut: Building Abstractions in Class Models: Formal Concept Analysis in a Model-Driven Approach. MoDELS 2006: 513-527

- UML Use case diagram refactoring

  * X. Dolques, M. Huchard, C. Nebut, and P. Reitz. Fixing generalization defects in UML use case diagrams. CLA 2010: 247-258

- Blob design defect correction

  * N. Moha, M. Rouane-Hacene, P. Valtchev, Y.-G. Guéhéneuc: Refactorings of Design Defects Using Relational Concept Analysis. ICFCA 2008: 289-304

- Extracting architectures in object-oriented software

  * A.-E. El Hamdouni, A. Seriai, M. Huchard Component-based Architecture Recovery from Object-Oriented Systems via Relational Concept Analysis. CLA 2010: 259-270

Applications

- Learning model Transformation patterns in MDE

  * X. Dolques, M. Huchard, and C. Nebut. From transformation traces to transformation rules: Assisting model driven engineering approach with formal concept analysis. In Supplementary Proc. of ICCS 2009:15-29.

- Classification of web services

  * Z. Azmeh, M. Driss, F. Hamoui, M. Huchard, N. Moha, C. Tibermacine, Selection of Composable Web Services Driven by User Requirements. To appear in the Application and Experience Track of ICWS 2011

- Ontology construction

  * R. Bendaoud, M. Rouane-Hacene, Y. Toussaint, B. Delecroix, and A. Napoli, Text-based ontology construction using relational concept analysis. MCETECH 2008

- Ontology pattern extraction

  * M. Rouane-Hacène, M. Huchard, A. Napoli, P. Valtchev. Using Formal Concept Analysis for discovering knowledge patterns. CLA 2010: 223-234

- Ontology restructuring

  * M. Rouane-Hacene, R. Nkambou, P. Valtchev. Supporting ontology design through large-scale FCA-based ontology restructuring, to appearin Proc. of the ICCS 2011.

A synthesis on RCA

- an iterative method to produce abstractions
- variations on scaling operators: $\exists, \forall, \forall\exists, \geq\ n\ r : c$, etc. (on relational contexts and on steps)
- object-attribute concept posets can be built instead of lattices to limit the complexity
- opportunities for enhancing application of FCA to software engineering domain

Tools

- Galicia: http://galicia.sourceforge.net/
- eRCA: http://code.google.com/p/erca/

Outline

Context and Motivations

Context

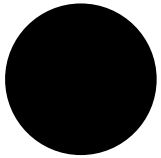
- Model driven development
- Development of a model transformation
- source and target metamodels require domain experts

Motivations

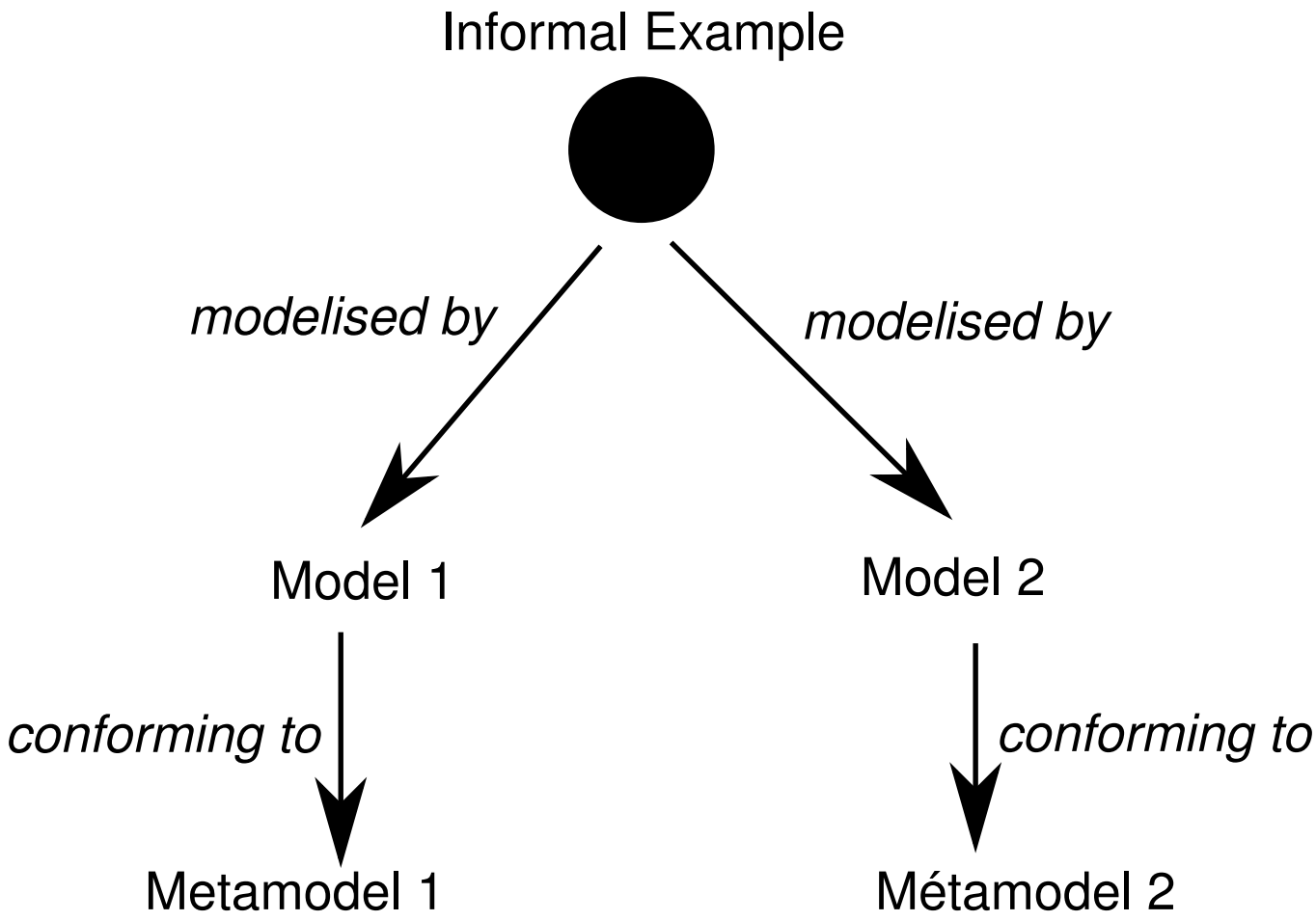
- Ease and speed up the development process of model transformations
- Improve the integration of domain expert in the process

The "By Example" approach

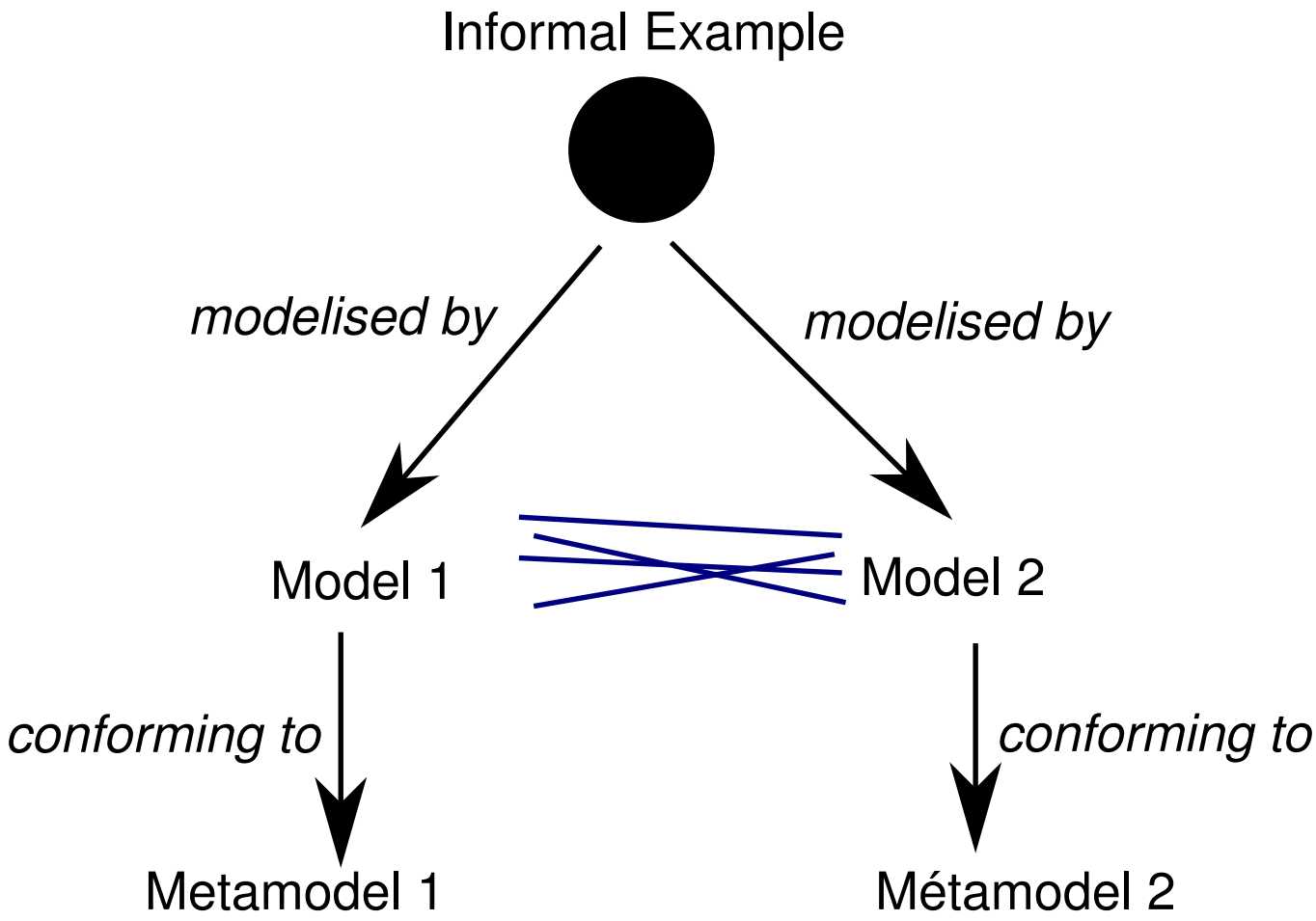
Informal Example



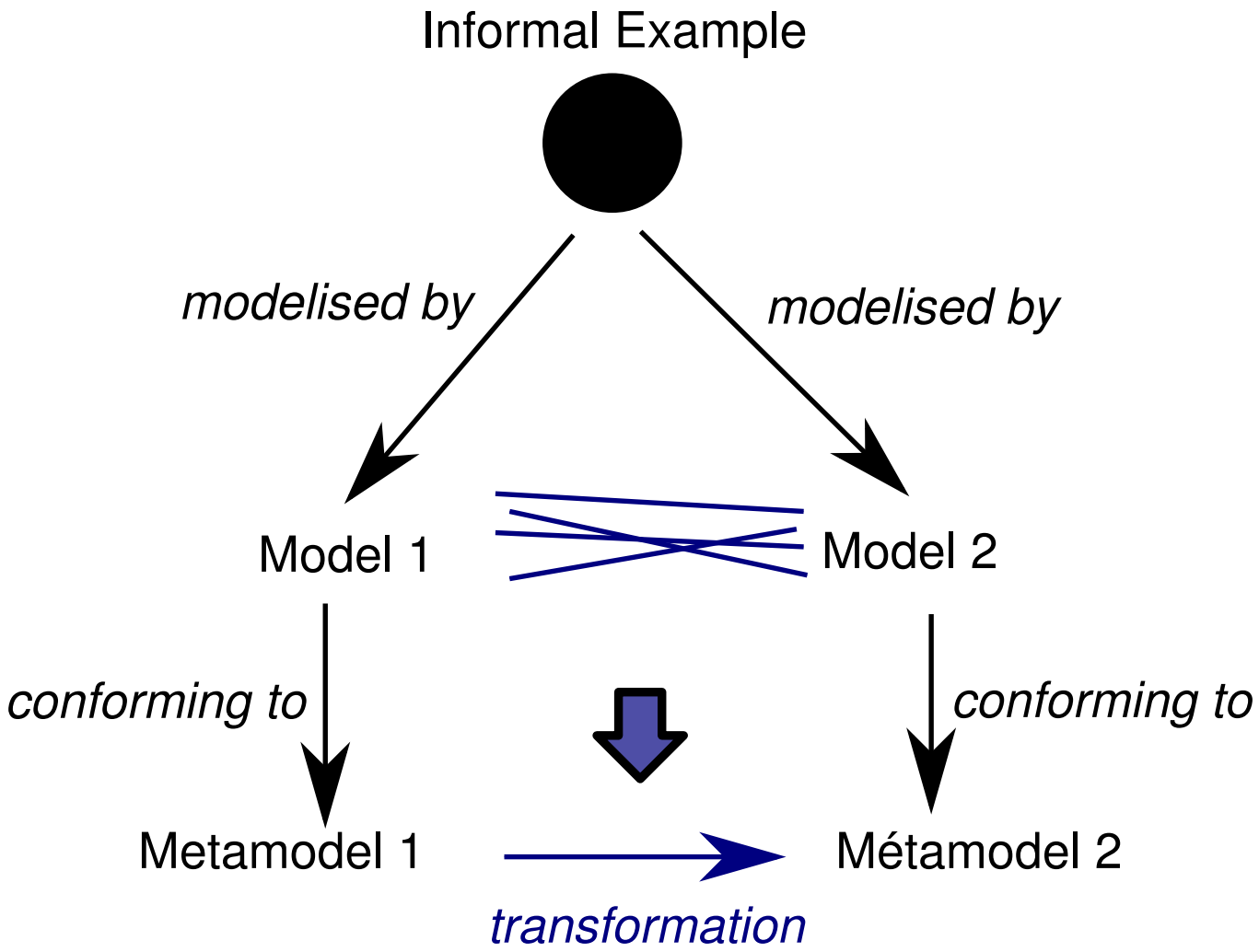
The "By Example" approach



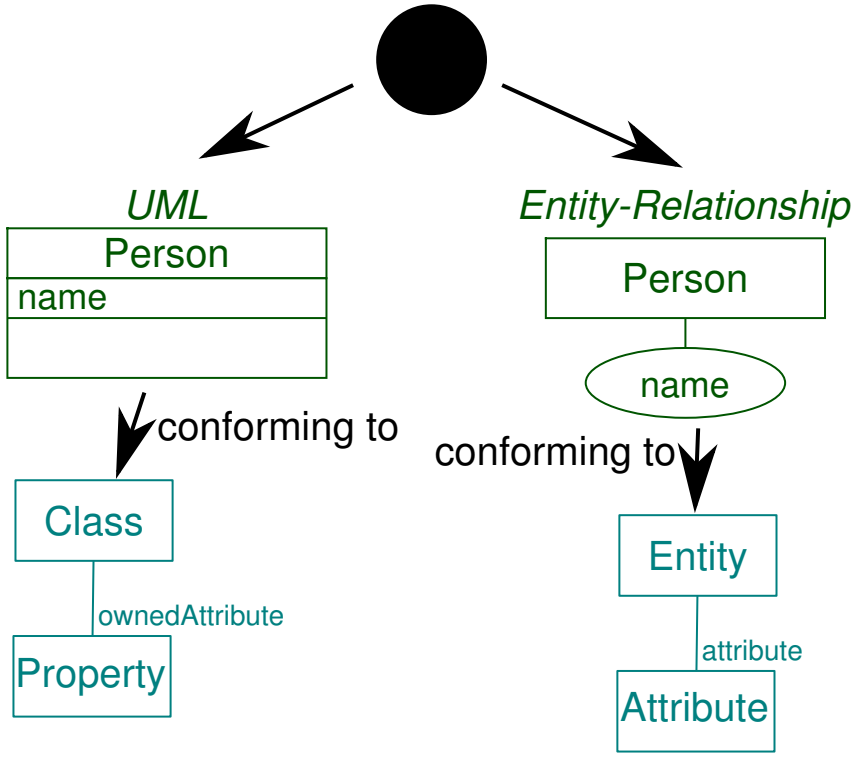
The "By Example" approach



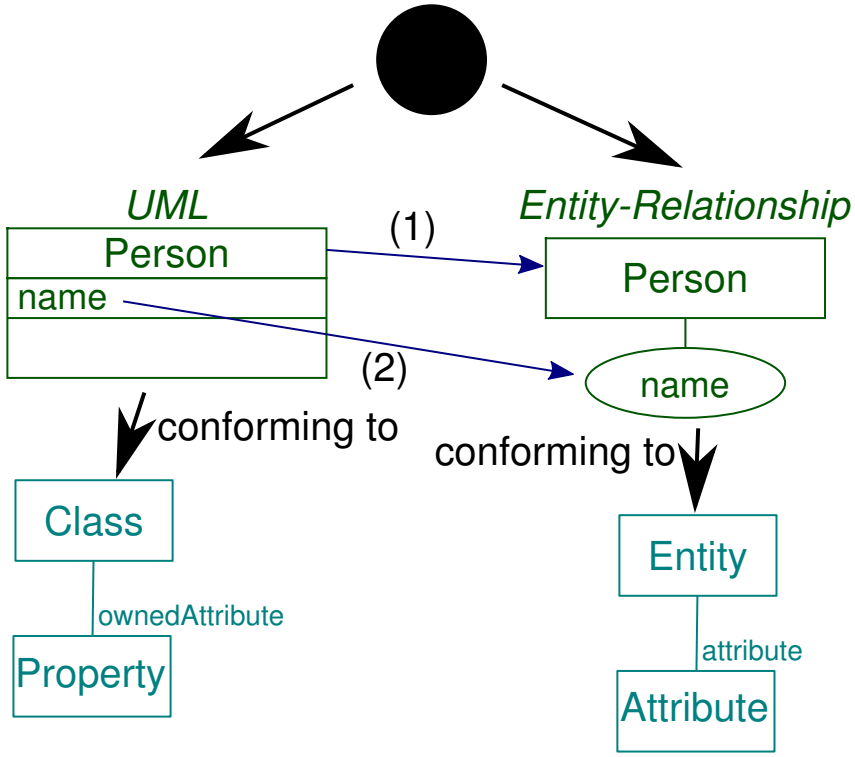
The "By Example" approach



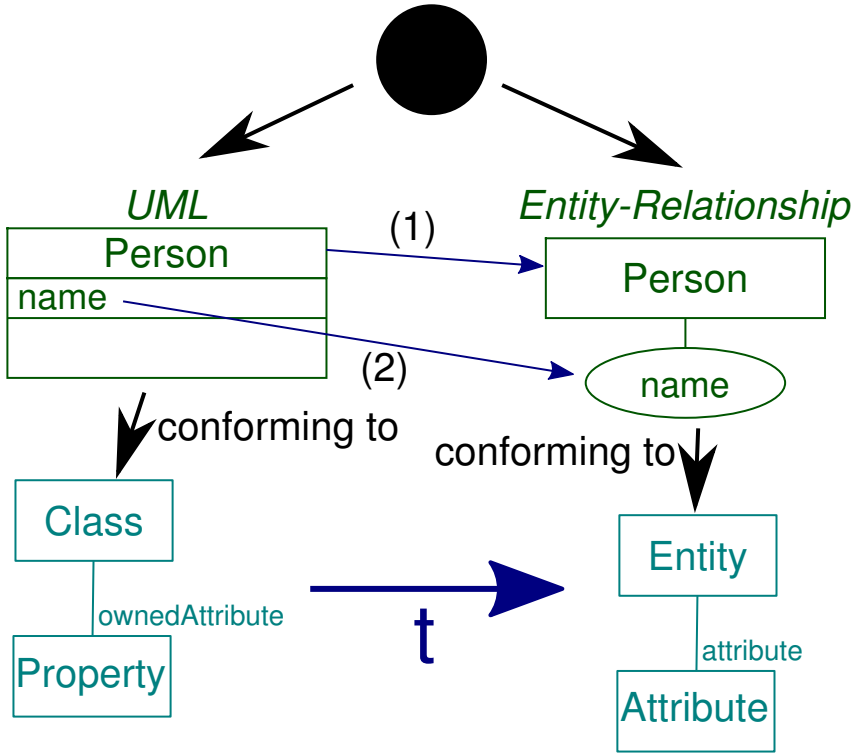
The "By Example" approach



The "By Example" approach



The "By Example" approach



(1) $instance(Class, x) \wedge ownedAttribute(x) \neq \emptyset \Rightarrow instance(Entity, t(x)) \wedge (y \in ownedAttribute(x) \rightarrow t(y) \in attribute(t(x)))$

(2) $instance(Property, x) \wedge association(x) = \emptyset \Rightarrow instance(Attribute, t(x))$

"By Example" approaches

[Balogh et Varró(2009)]

- **Input matching**: set of typed couples of elements
- **Matching creation**: manually
- **Input specific development**: none
- **Learning principle**: Inductive Logic Programming
- **Output data**: transformation rules (VIATRA)

"By Example" approaches

[Wimmer et al.(2007)]

- **Input matching**: set of couples of elements
- **Matching creation**: manually
- **Input specific development**: explicit constraints of the transformation from concrete to abstract syntax
- **Learning principle** : *ad hoc* method
- **Output data** : ATL code

"By Example" approaches

[Kessentini et al.(2008)]

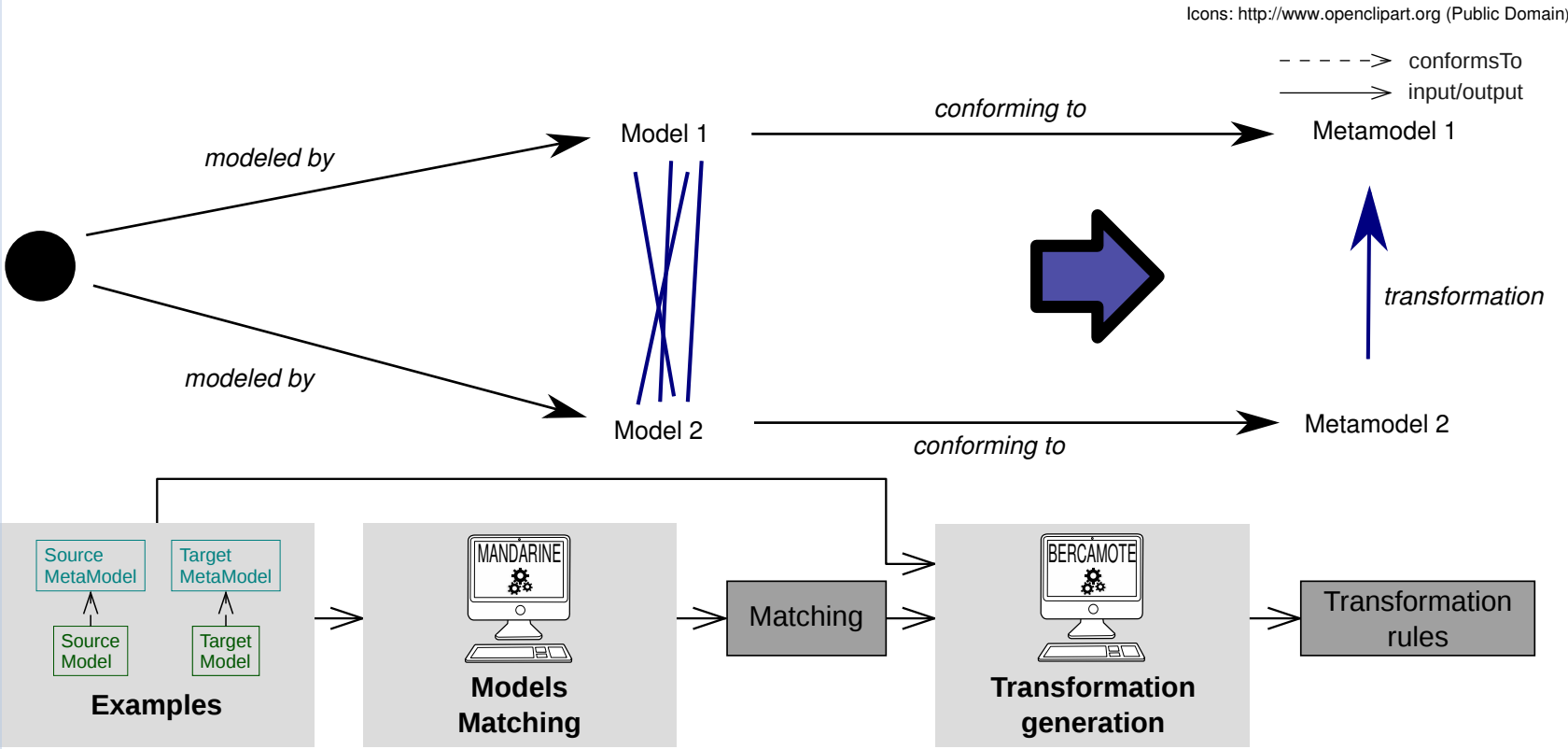
- **Input matching**: set of block couples
- **Matching creation**: manually
- **Input specific development**: none
- **Learning principle** : metaheuristics
- **Output data**: a transformed model

"By Example" approaches

Our approach

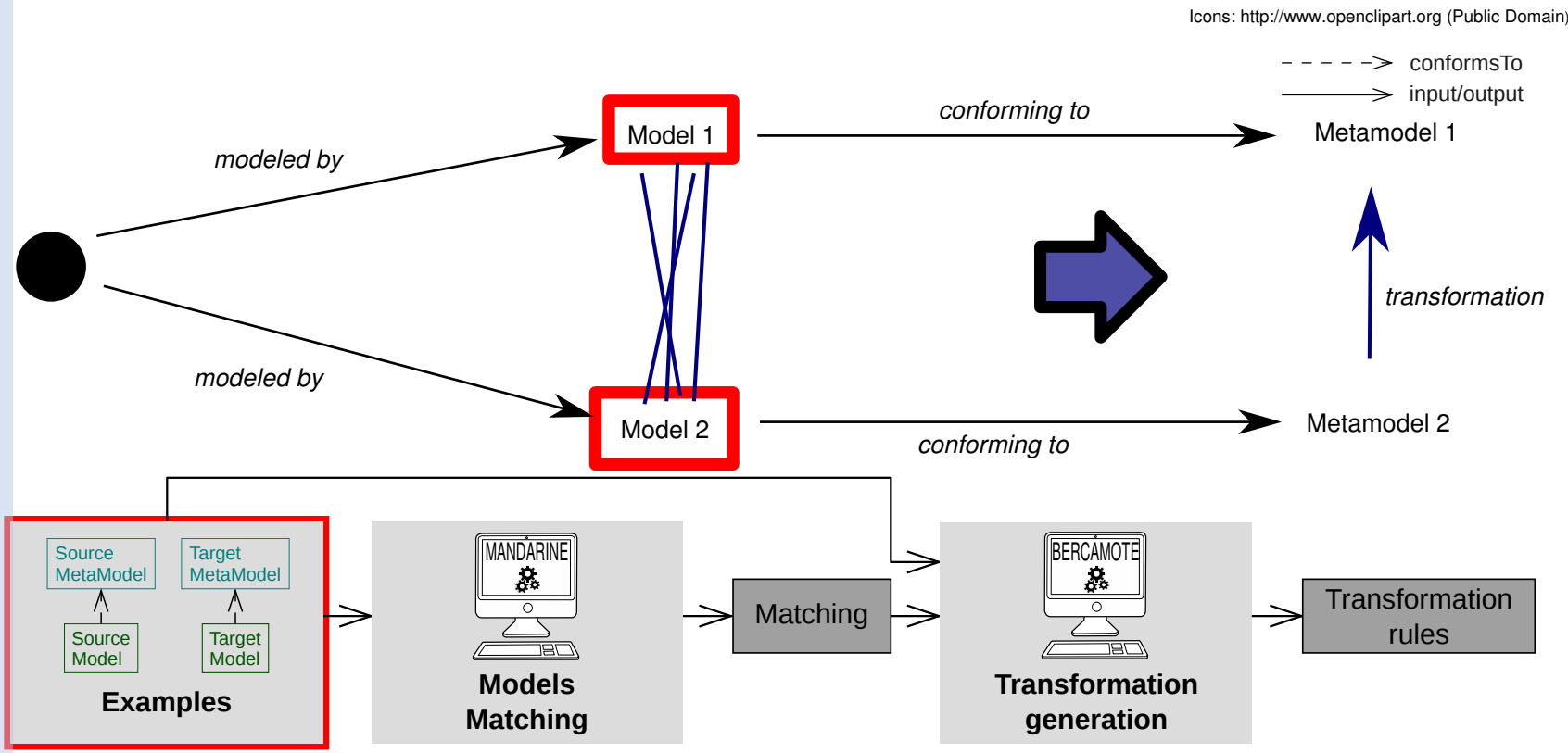
- **Input matching**: set of couples of elements
- **Matching creation**: matching assisted by tool
- **Input specific development**: none
- **Learning principle** : Relational Concept Analysis
- **Output data**: specification of transformation rules ordered in a lattice

General approach

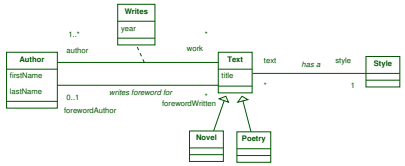


Outline

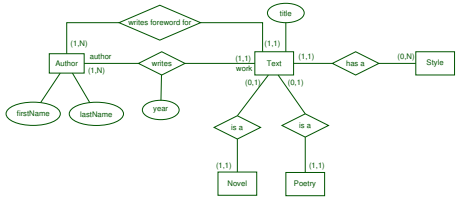
Illustrative Example



Tranformation Example



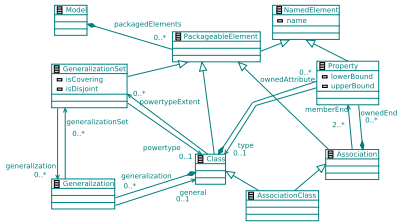
UML model



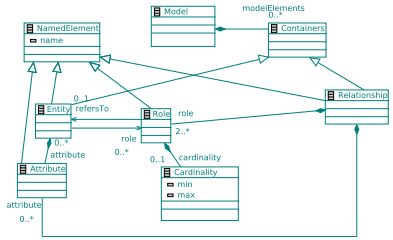
Entity-Relationship model

conforming to

conforming to

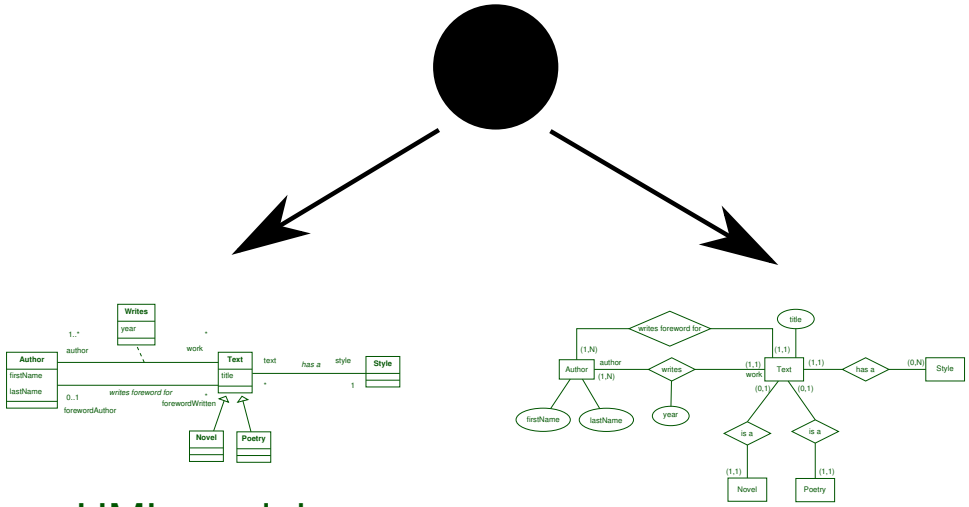


UML metamodel



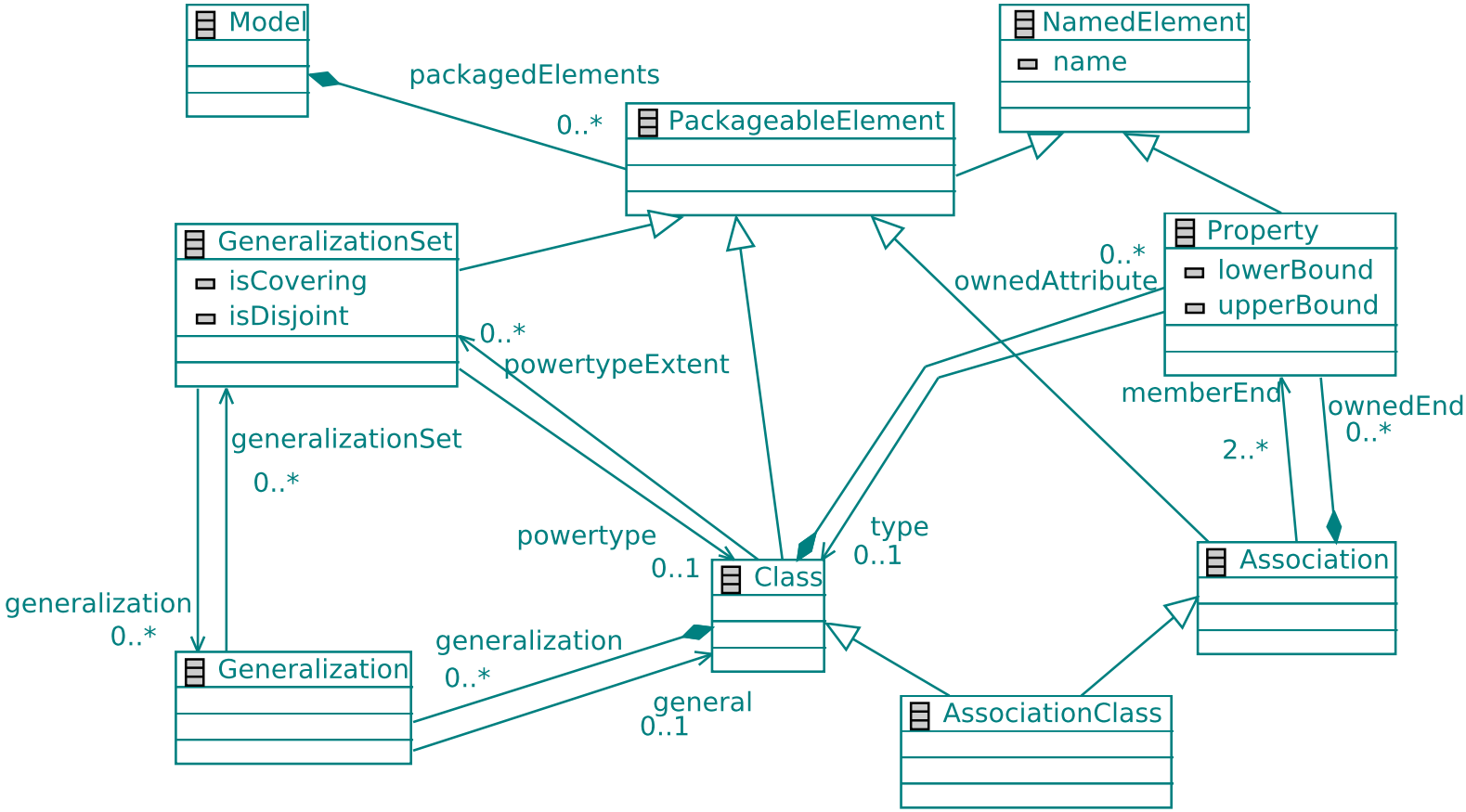
Entity-Relationship metamodel

Metamodels involved in the transformation



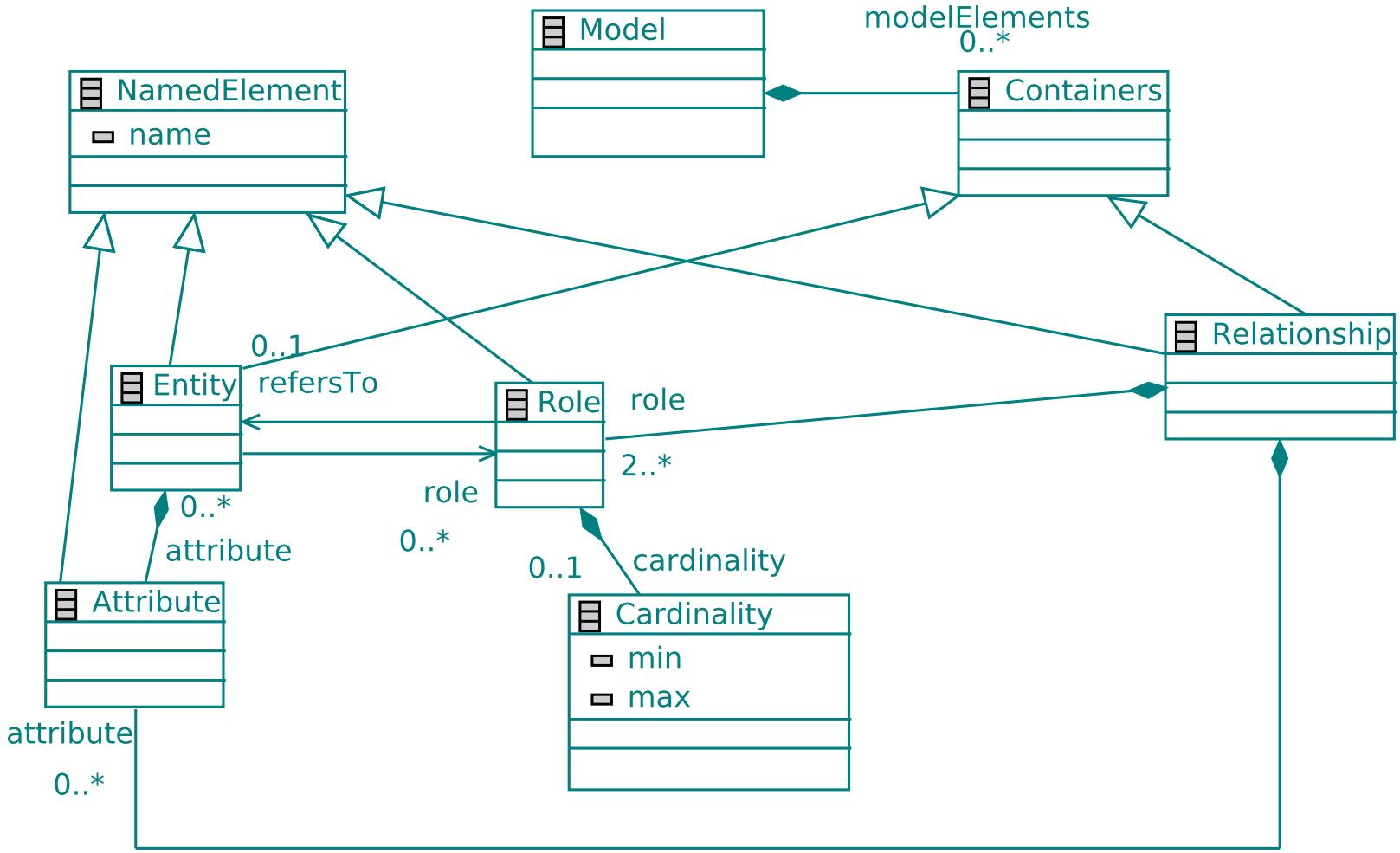
Metamodels involved in the transformation

Excerpt of UML metamodel

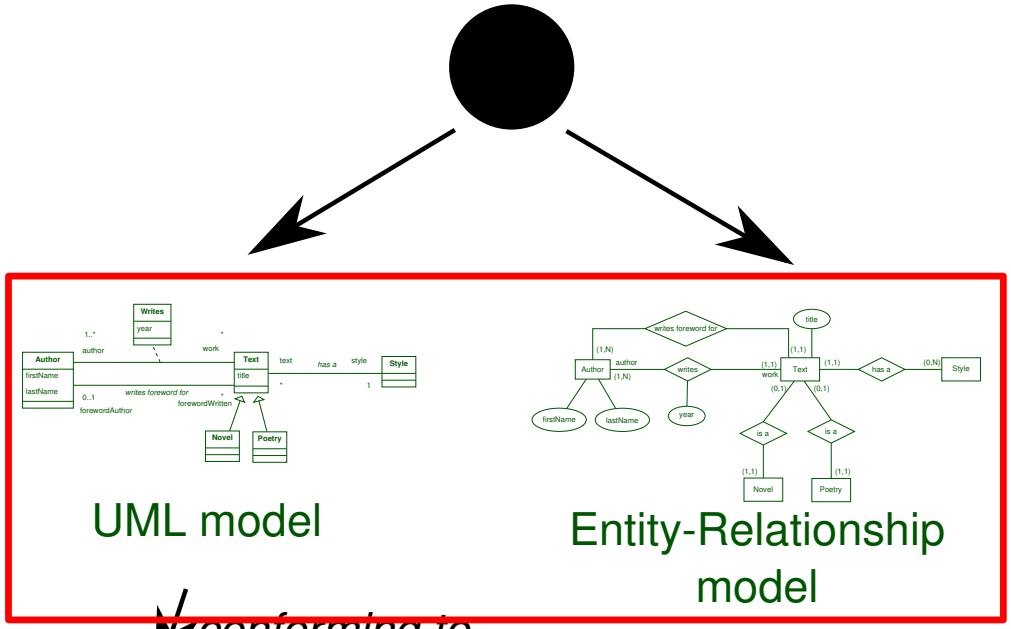


Metamodels involved in the transformation

Entity-Relationship metamodel

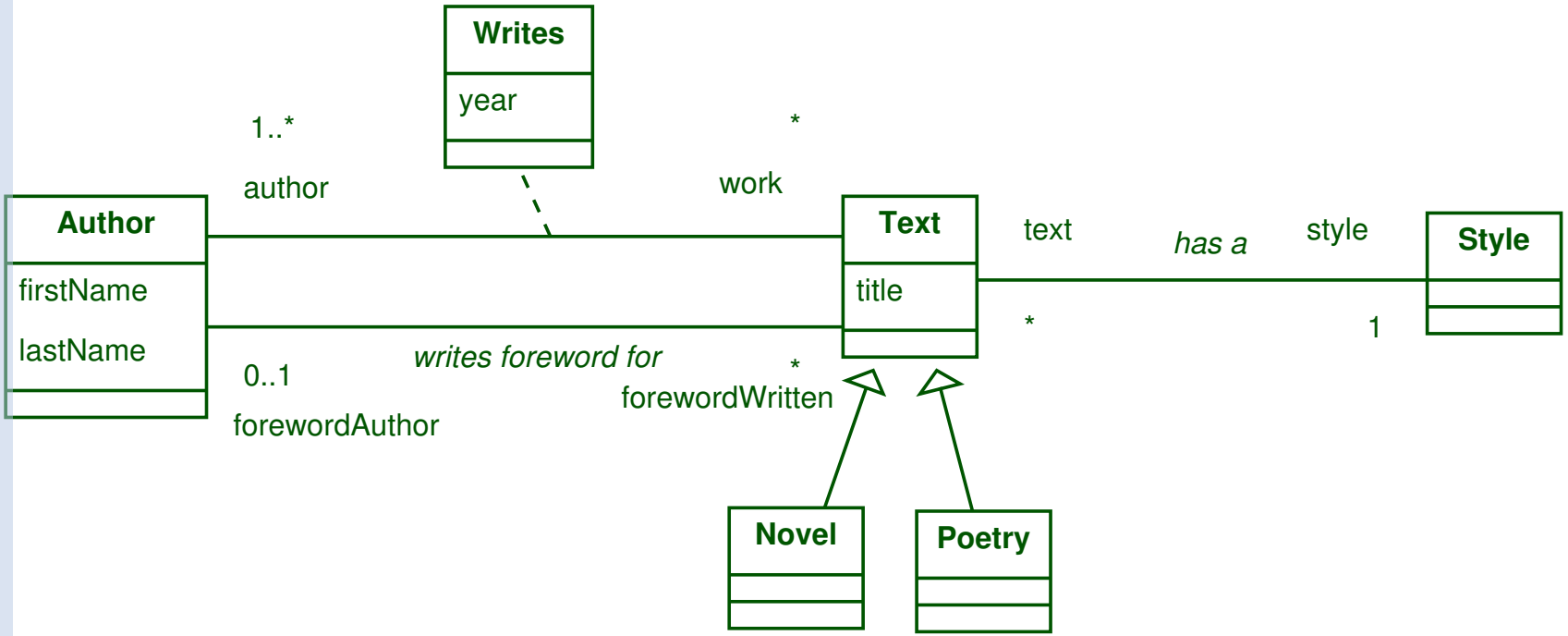


Models involved in the transformation



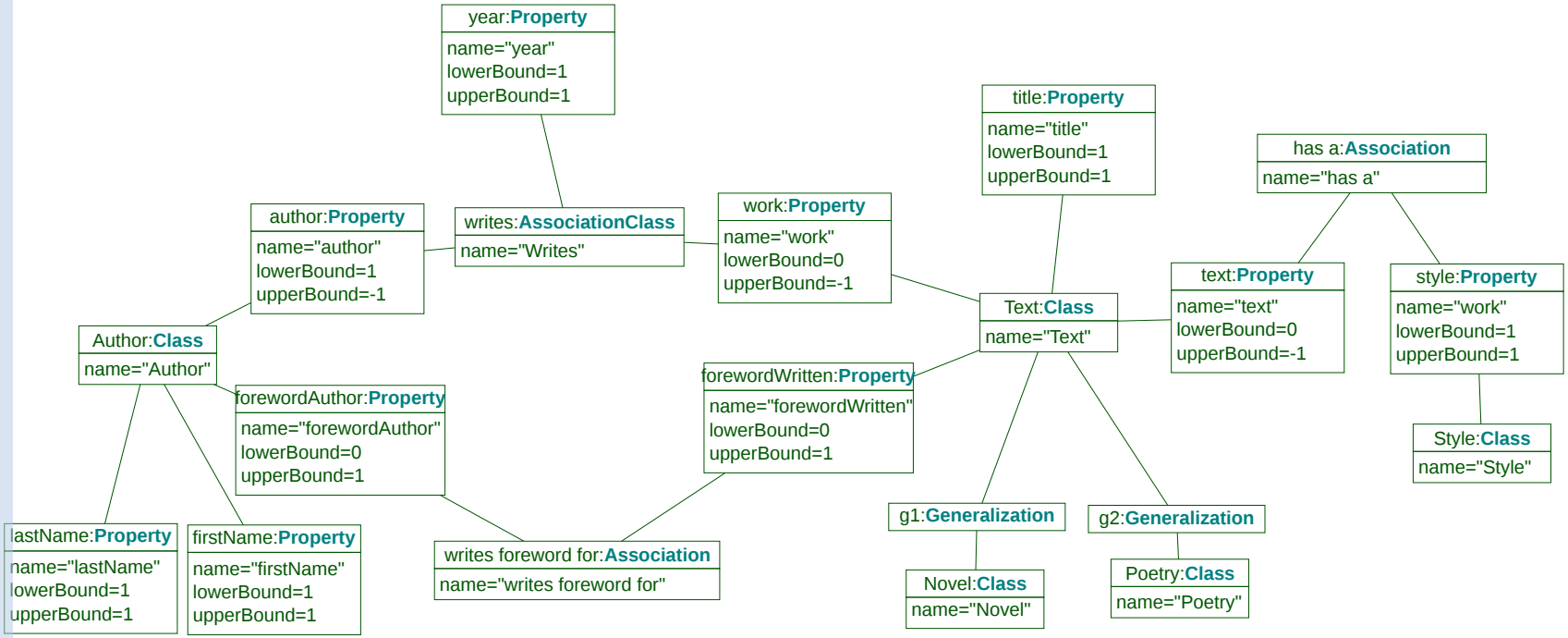
Models of our Example

A UML Model in concrete syntax



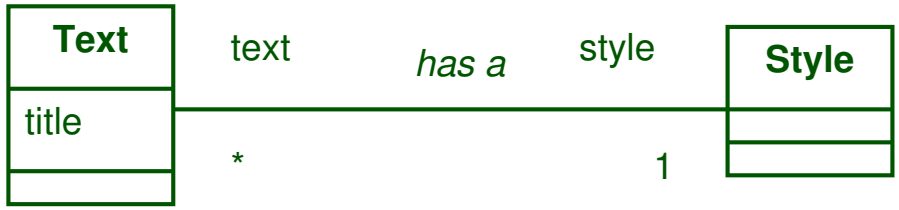
Models of our Example

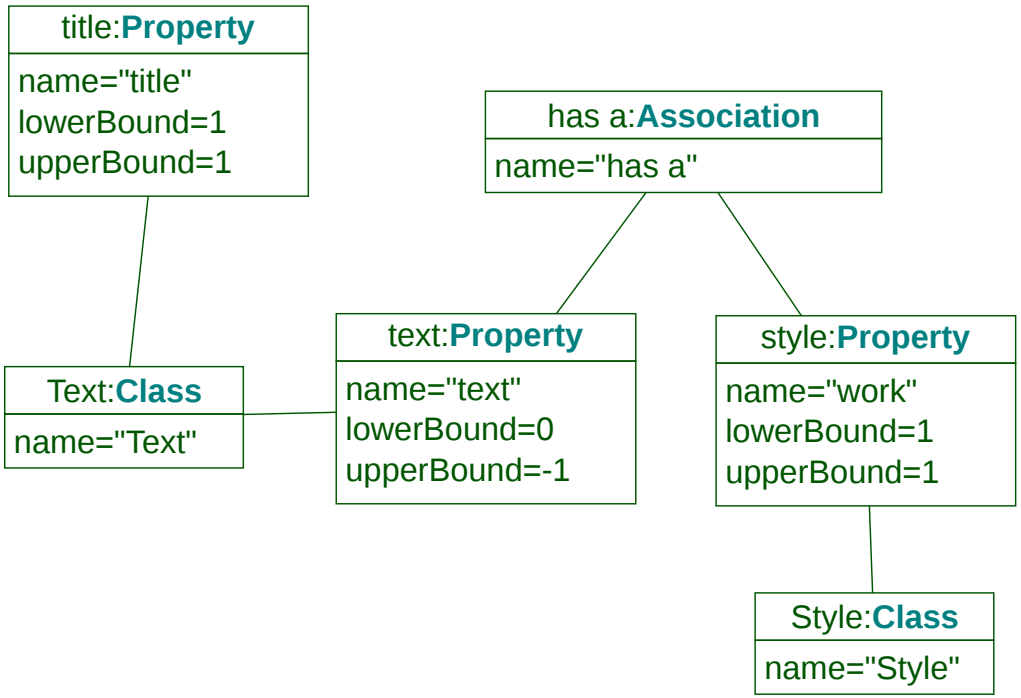
A UML Model in abstract syntax



Models of our Example

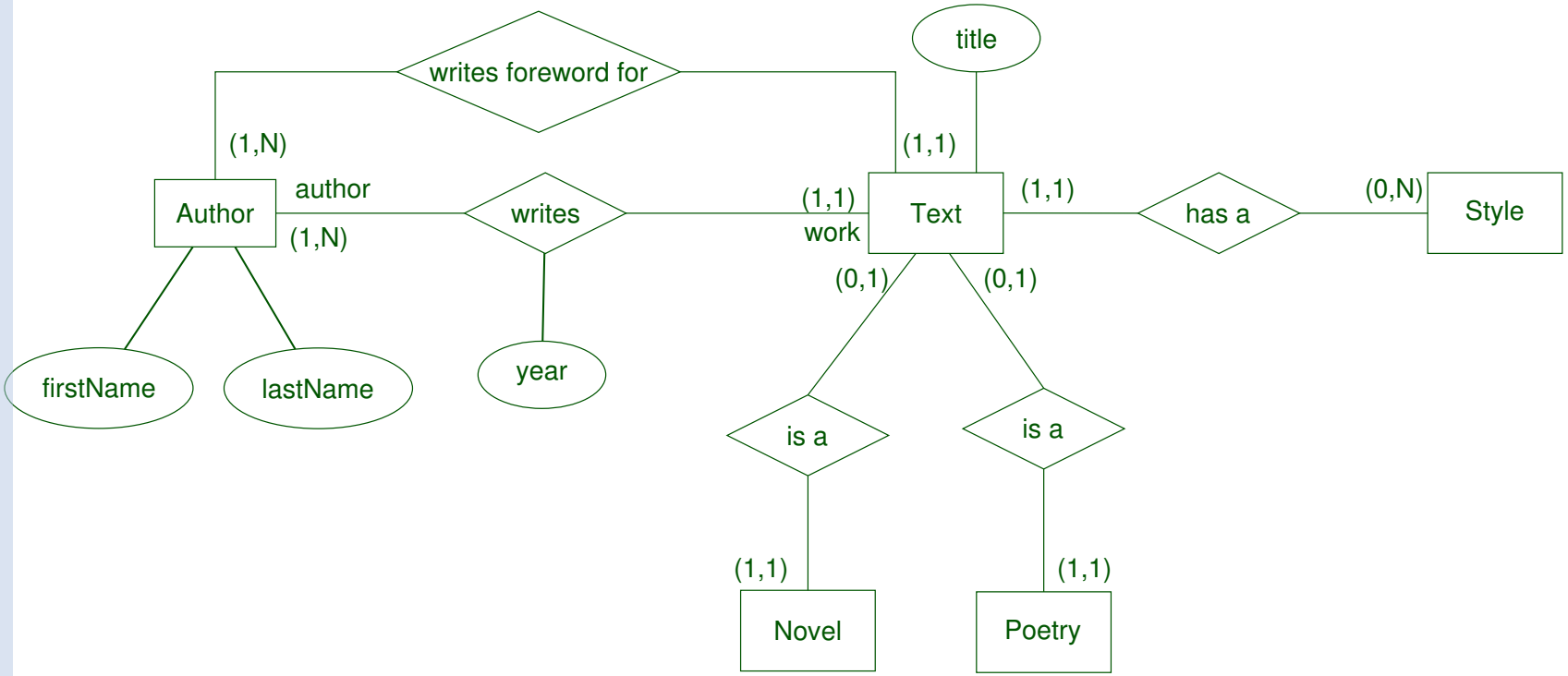
A UML model (excerpt)





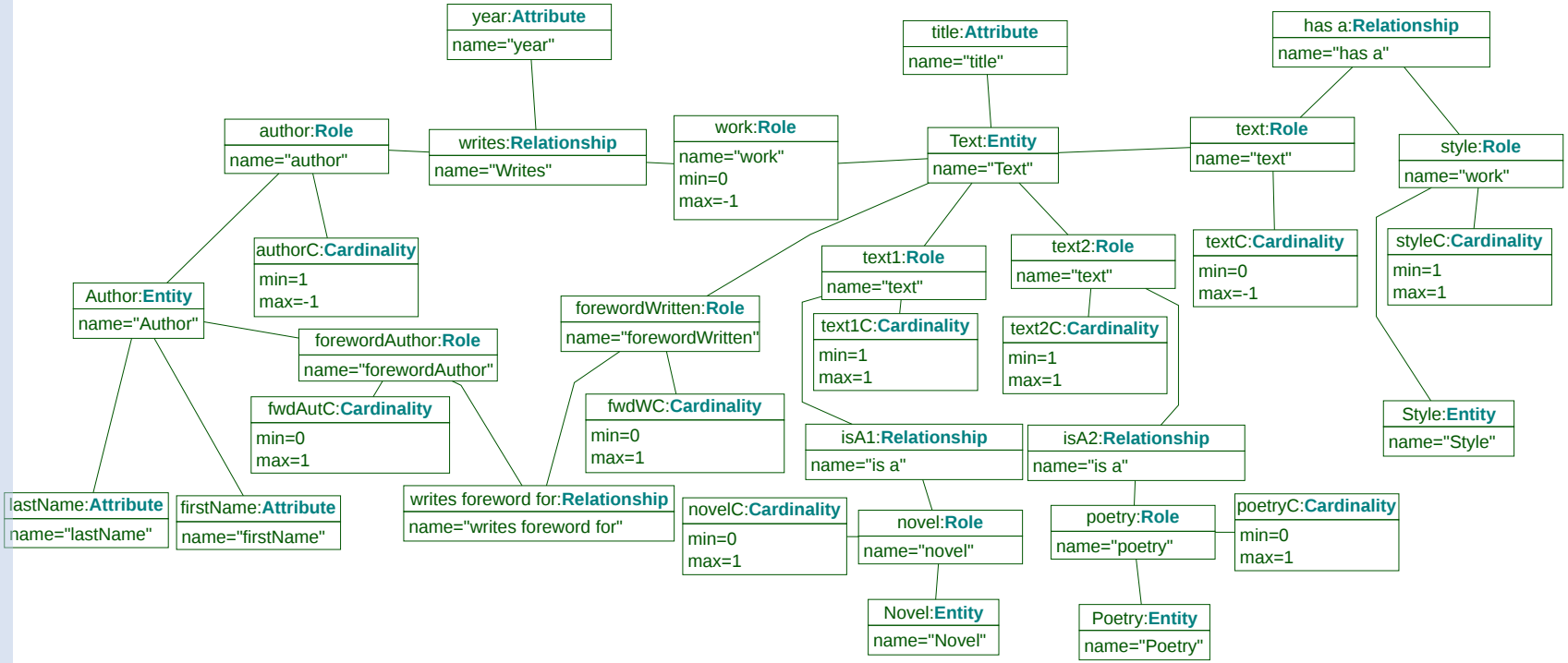
Models of our Example

An Entity-Relationship model in concrete syntax



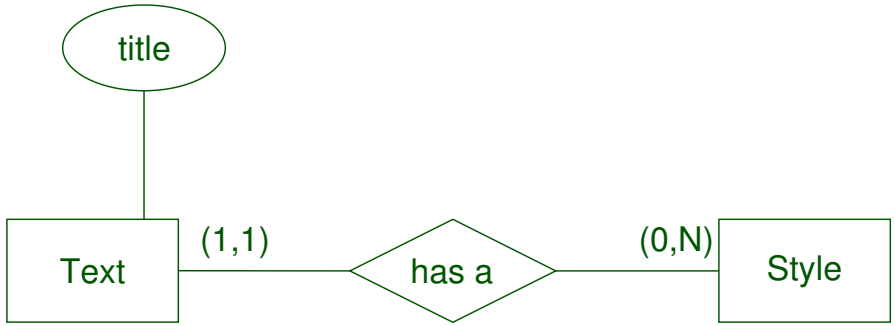
Models of our Example

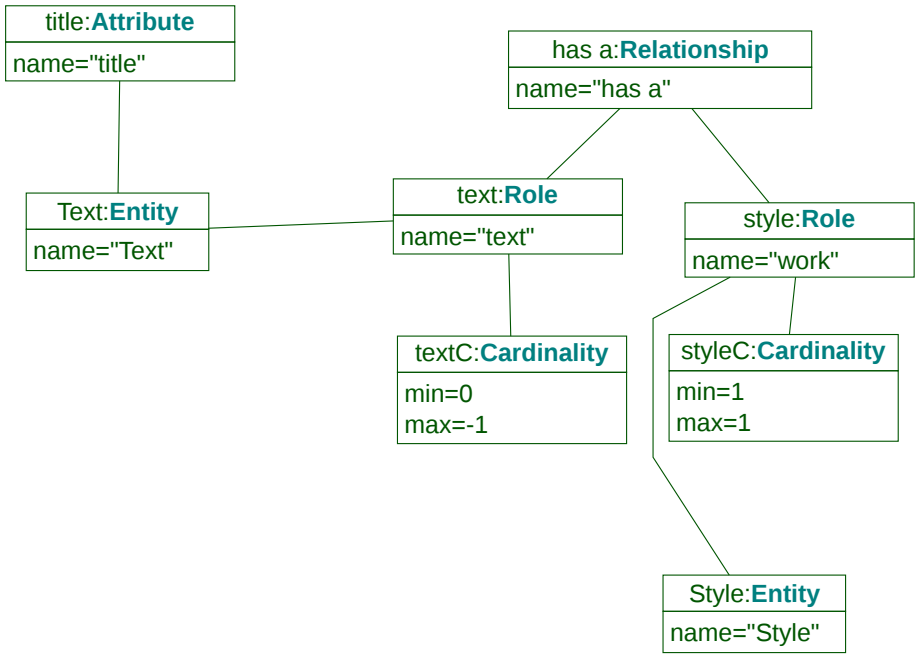
An Entity-Relationship model in concrete syntax



Models of our Example

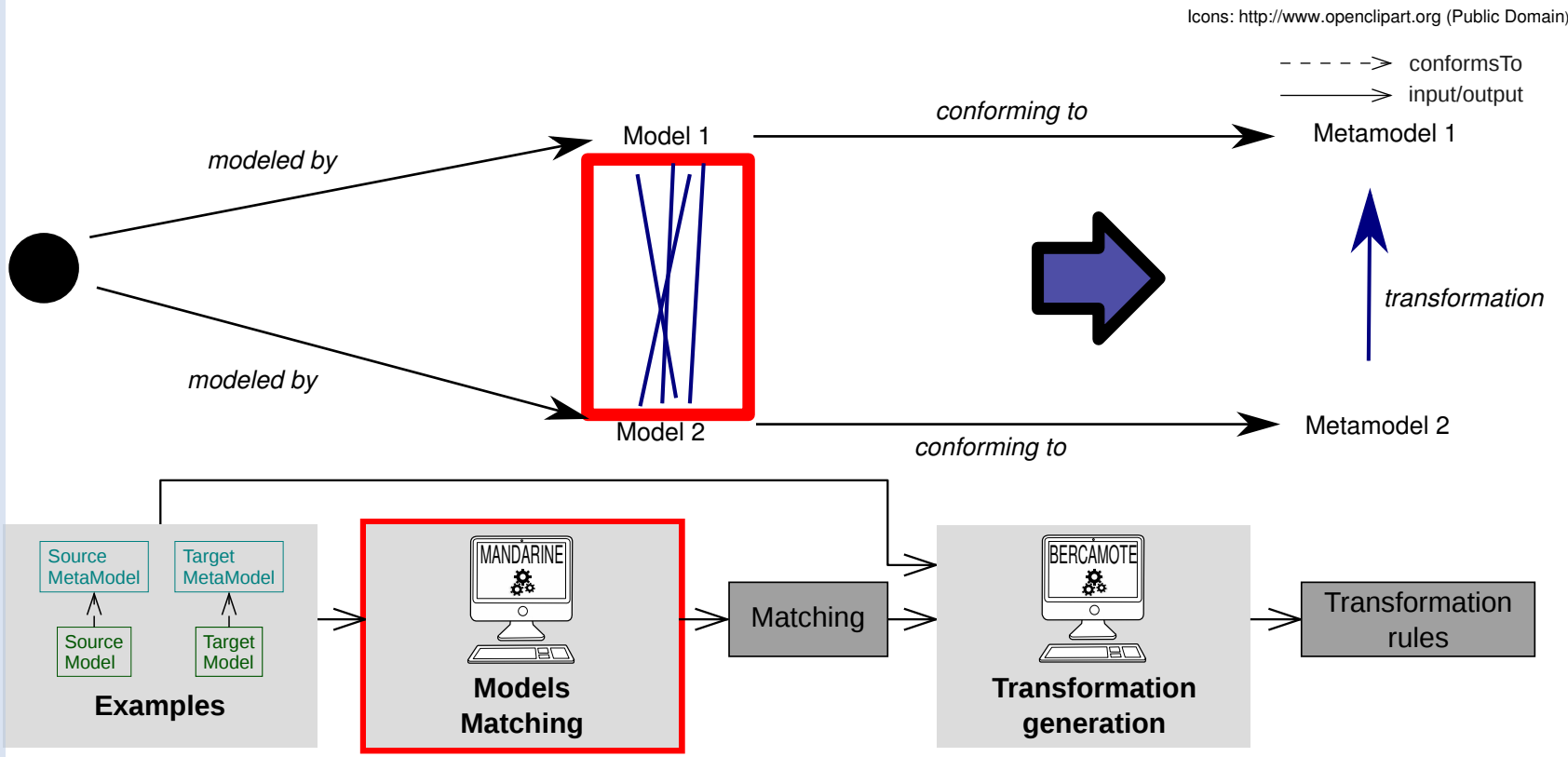
An Entity-Relationship model (excerpt)





Outline

Models Matching



Models Matching

Hypotheses

- the informal starting example contains named elements: those elements are to be found in the two models
- source and target models structure are close enough

State of the Art

Constraints of the matching methods that can be applied

- can be applied on a graph structure (not only a tree)
- not strictly based on semantic analysis

Relevant matching approaches

- Similarity Flooding [Melnik et al., 2002]
- OLA [Euzenat et al., 2004]
- **Anchor Prompt** [Noy et Musen, 2001]

Advantage of anchorPrompt

does not use similarity on relations names (relations names come from the metamodel level)

Description of process

Matching using Attributes values

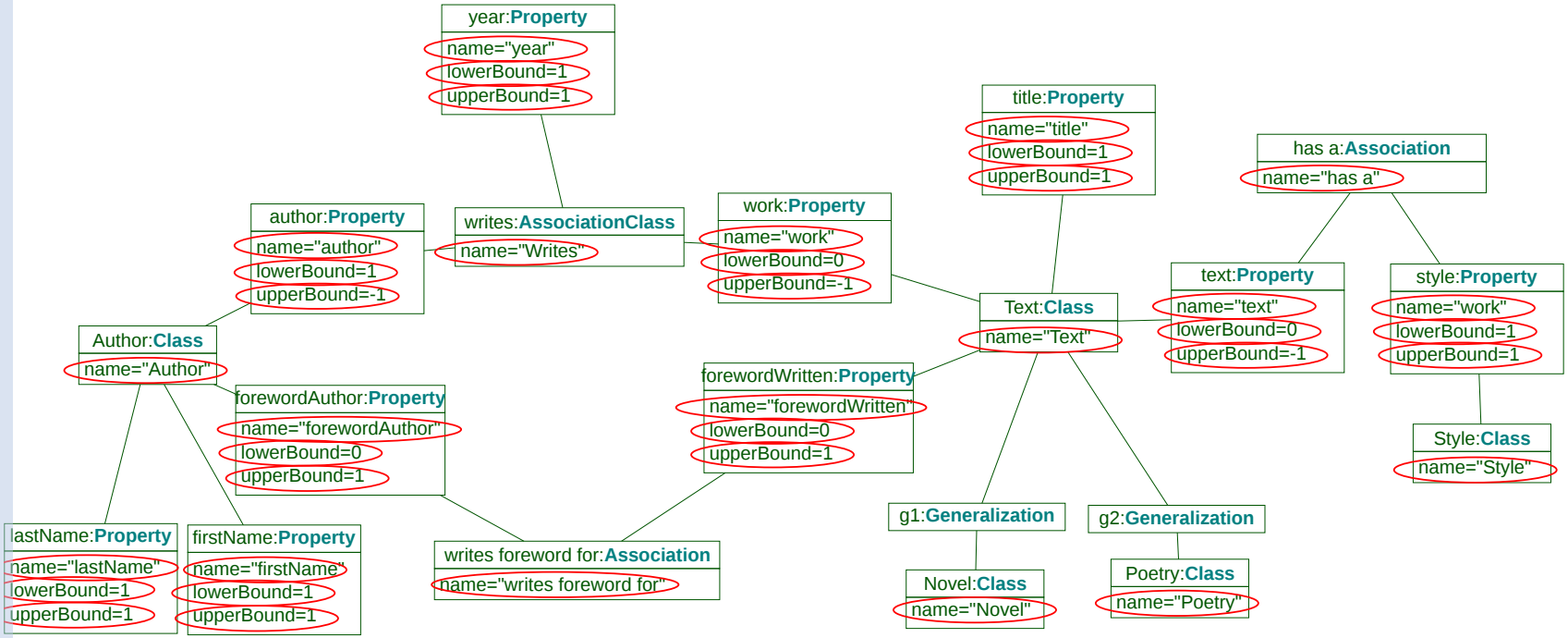
- enumeration of values in the models
- matching of similar values
- matching of the elements containing those values

Matching Propagation

- using structure similarity assumption

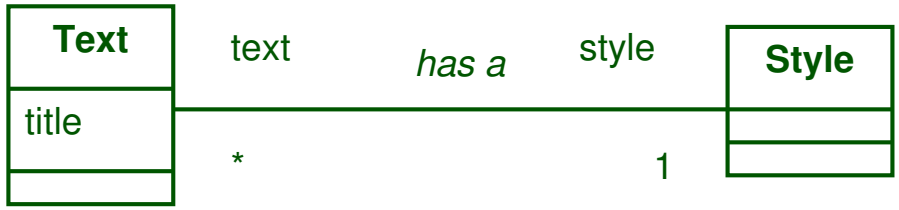
Attribute Instances

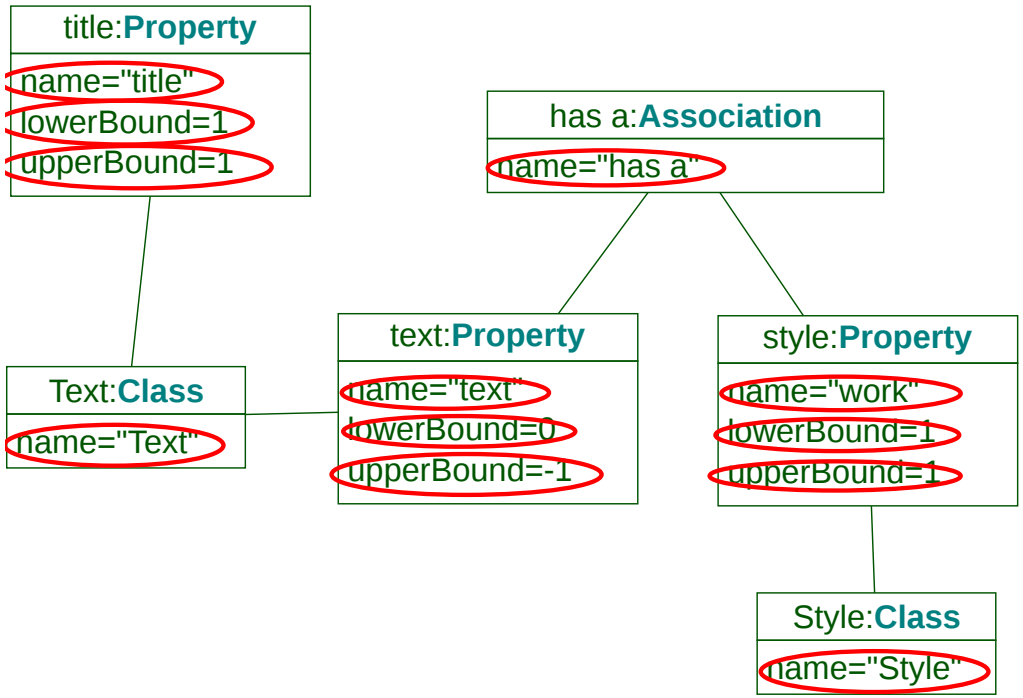
In the UML source model



Attribute Instances

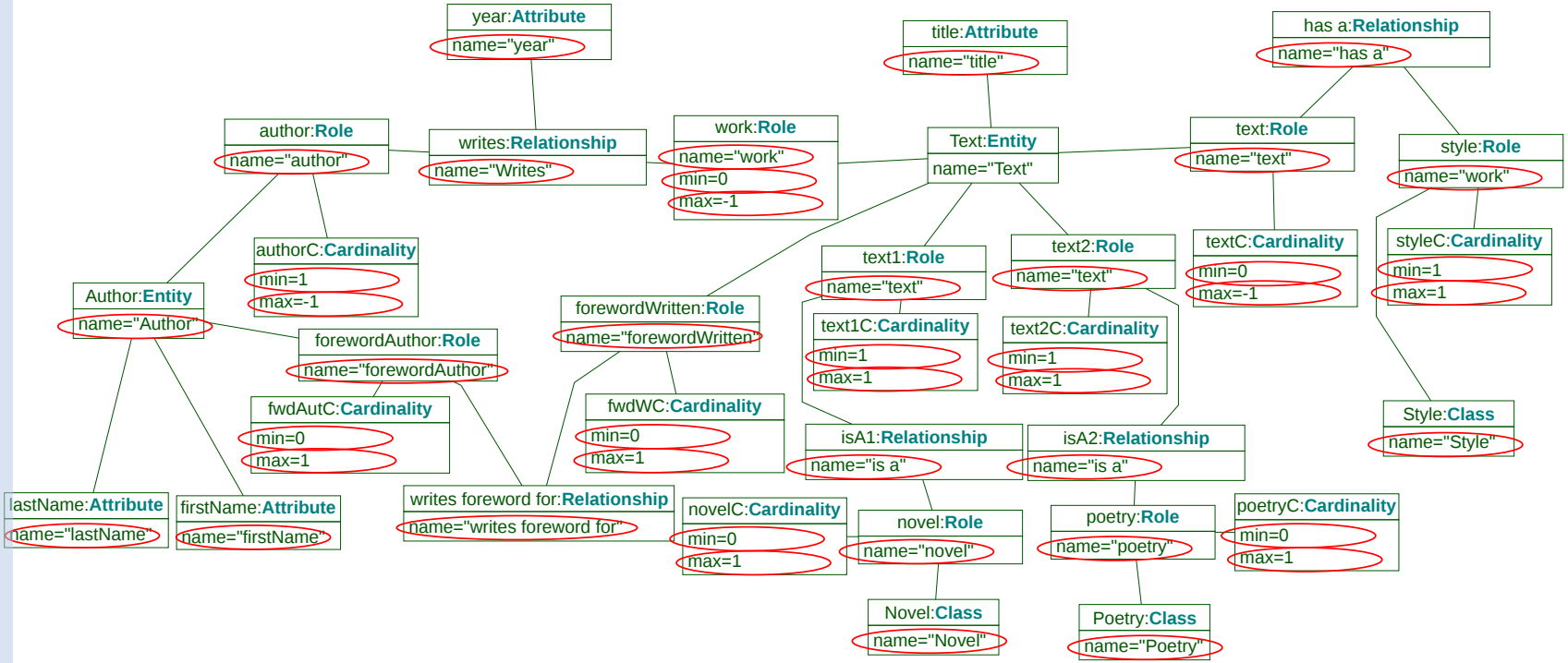
In the UML source model (excerpt)





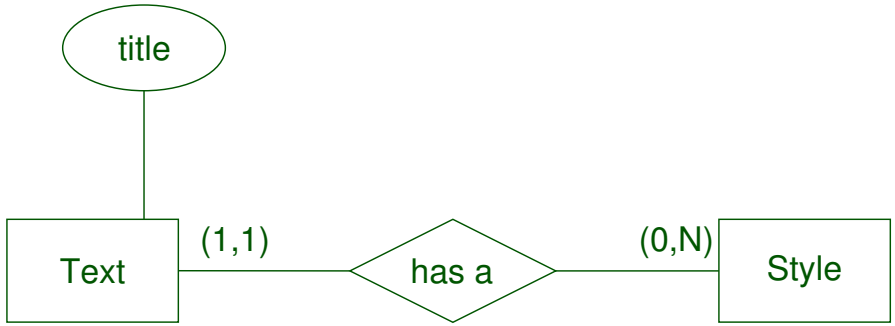
Attribut Instances

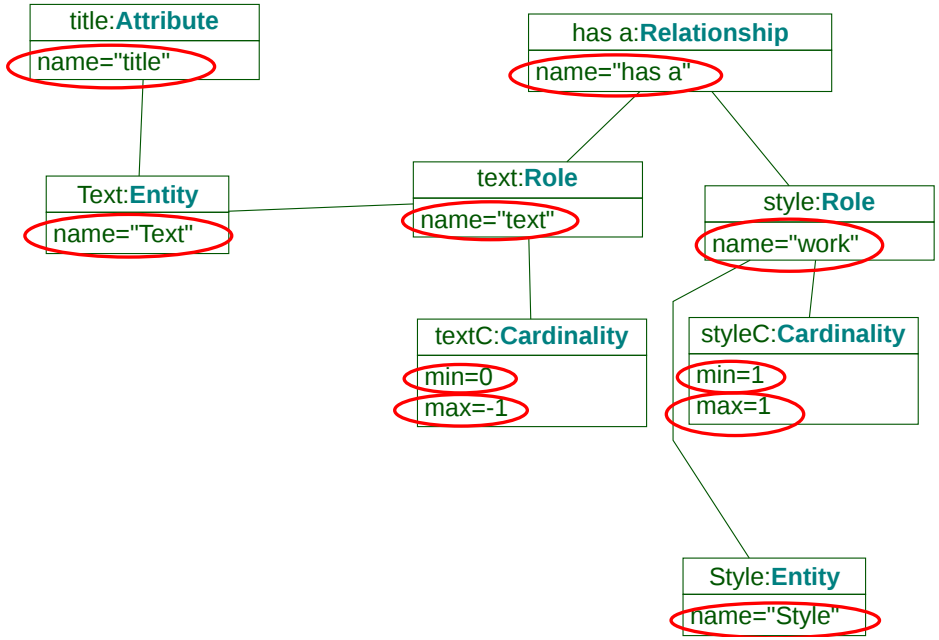
In the Entity-Relationship target model



Attribute Instances

In the Entity-Relationship target model (excerpt)





Attributes Instances

Compare source values to target values

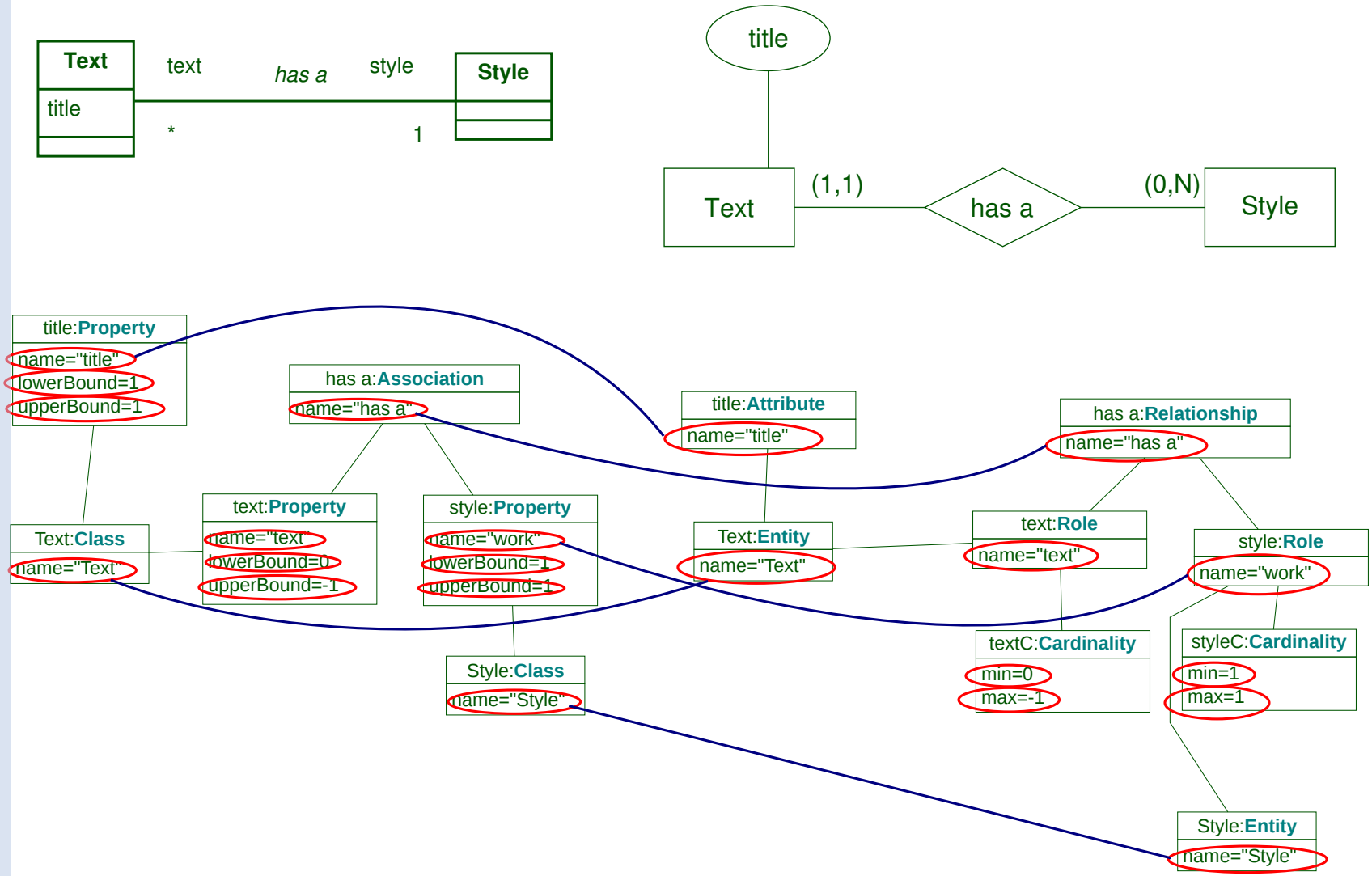
Value comparison

A source attribute instance and a target attribute instance match if all the following conditions are respected:

- they have the same value
- this value appears only once in the source model and the target model

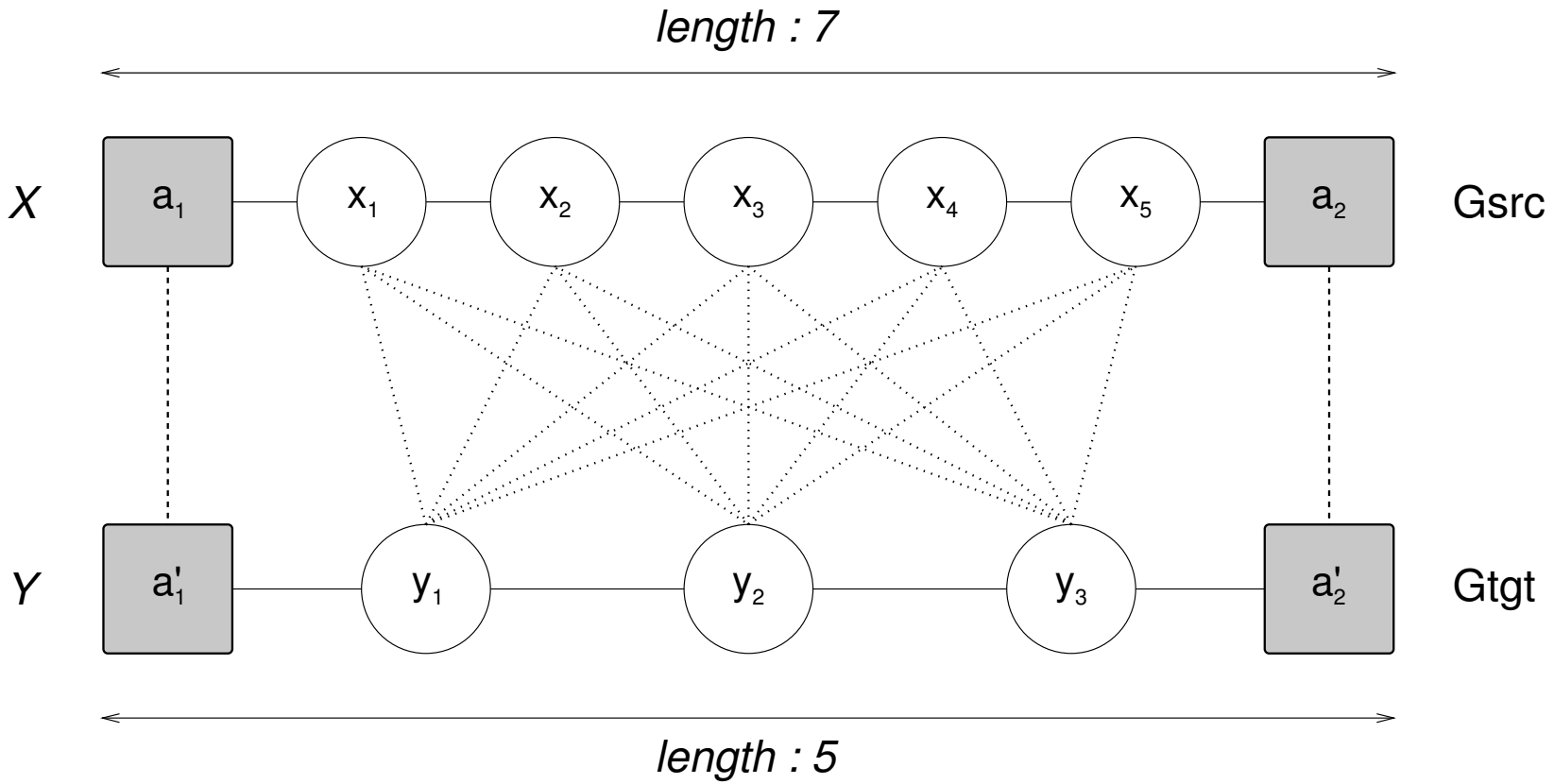
Value matchings in attributes

A first Model Matching



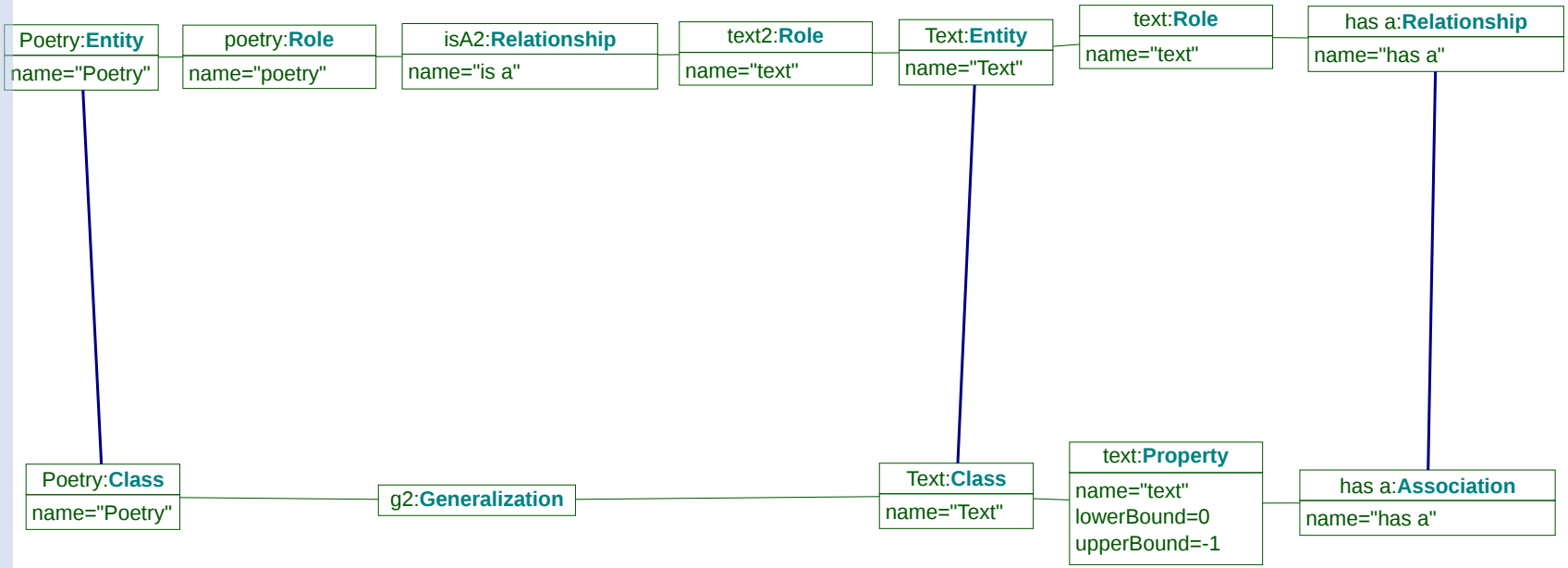
Propagation of the maching

The AnchorPrompt approach Adaptation

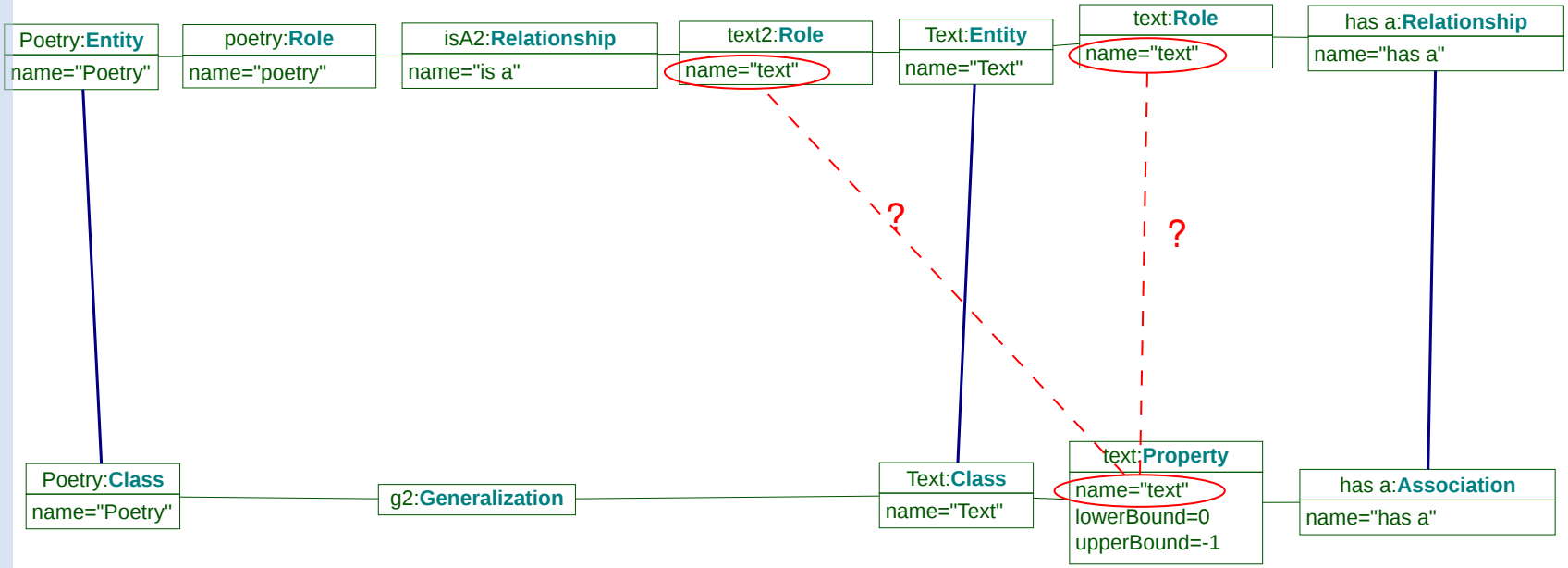


$$W(x,y) = 1 - \left| \frac{index(x)}{length(X)-1} - \frac{index(y)}{length(Y)-1} \right|$$

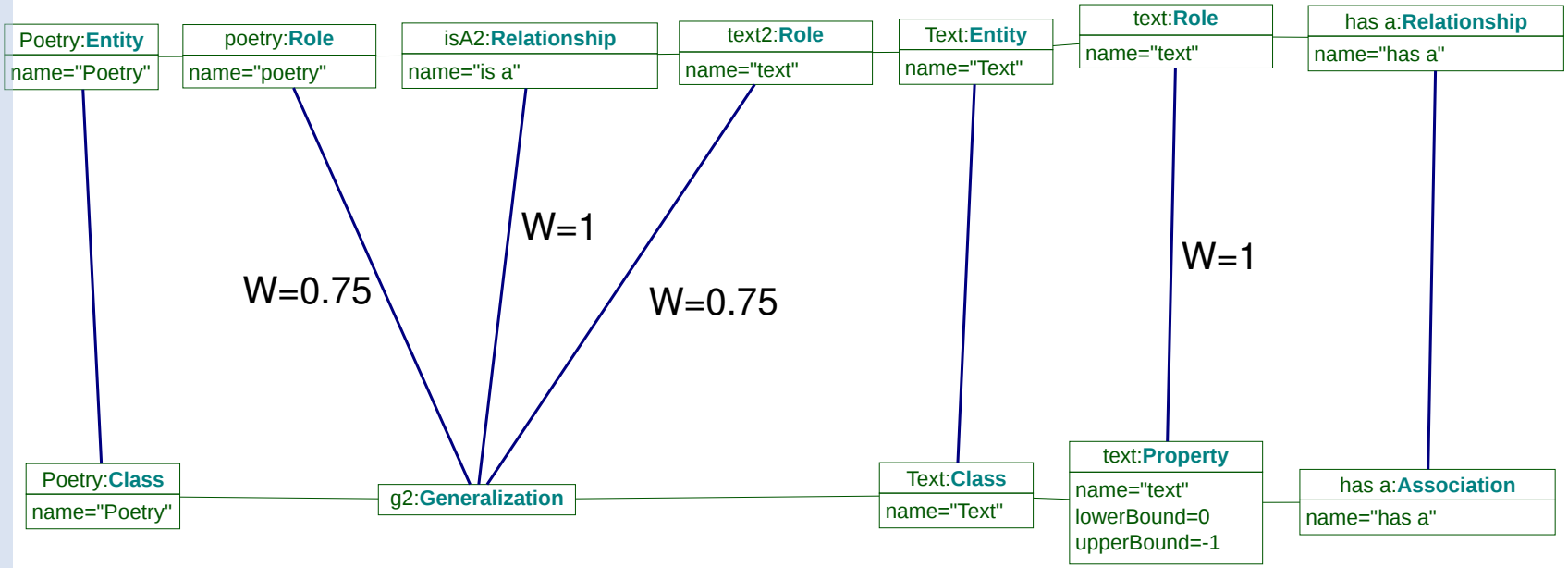
Similarities propagation



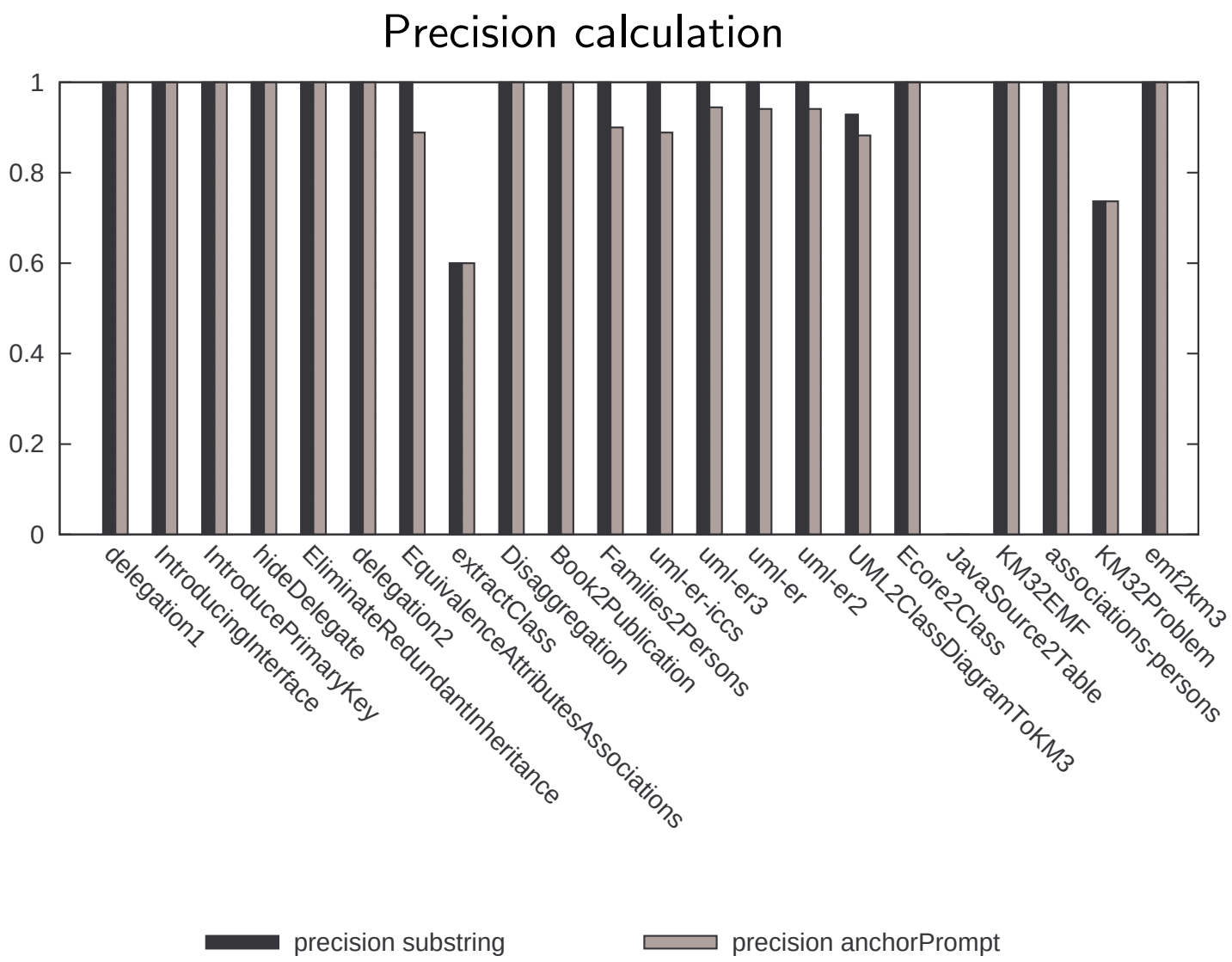
Similarities propagation



Similarities propagation



Empirical results



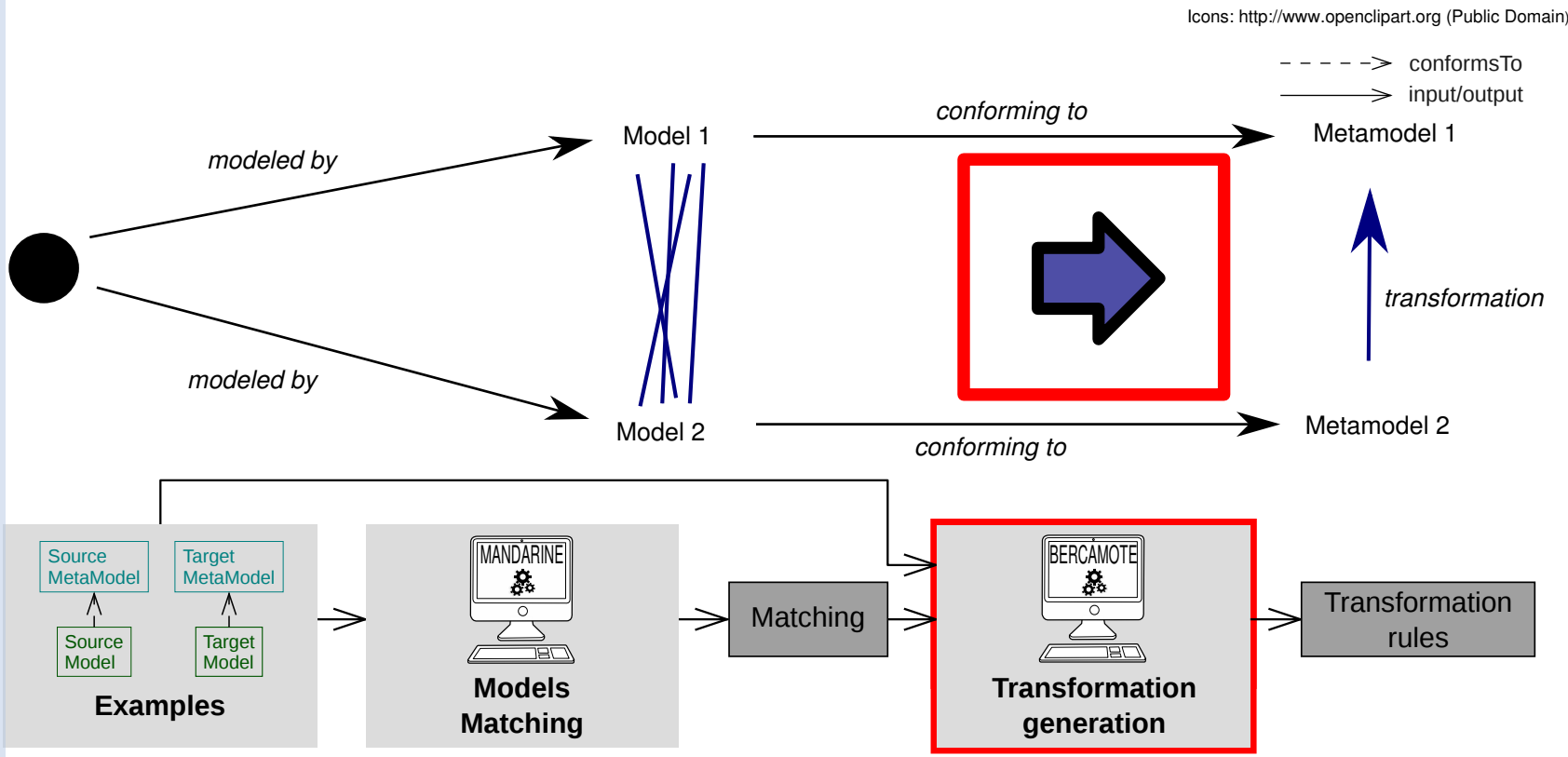
Synthesis

To address the model matching problem

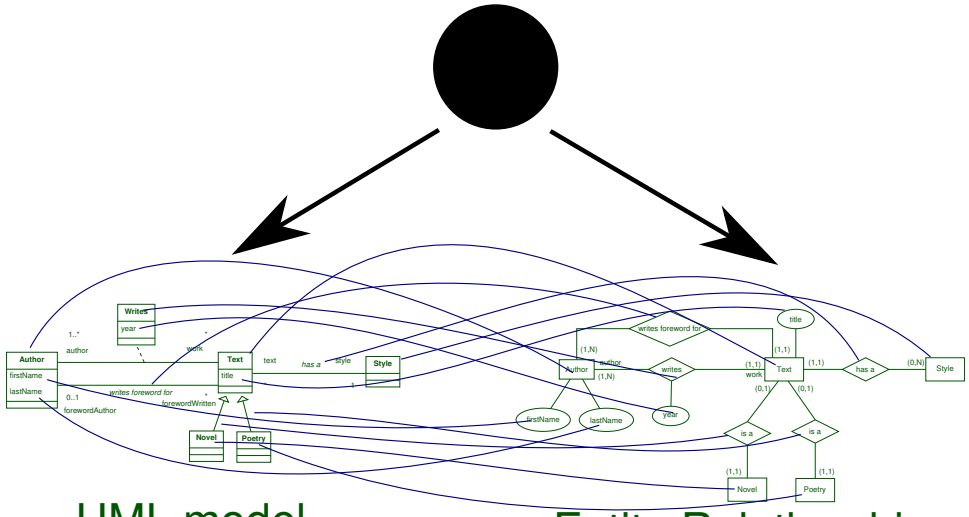
- Starting with 2 assumptions
  - the starting example contains named element
  - the models to match are structurally close
- We propose a process generating a model matching
  - using attributes values for a first matching with high confidence level
  - using an adaptation of AnchorPROMPT for extending the first matching

Outline

Transformation rules generation



Input Data

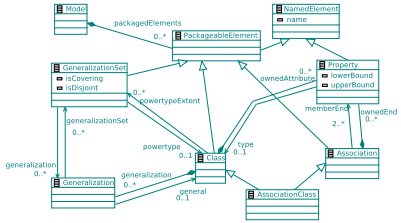


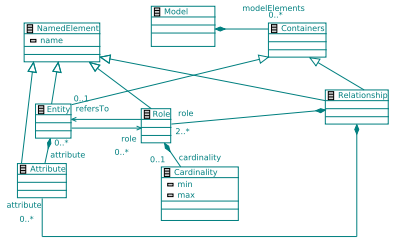
UML model

Entity-Relationship model

conforming to

conforming to

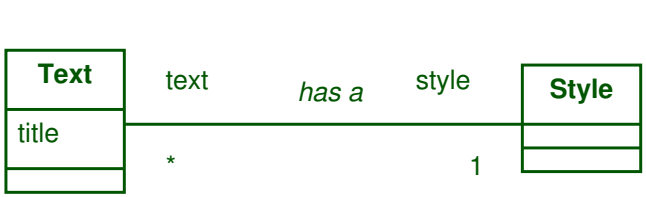


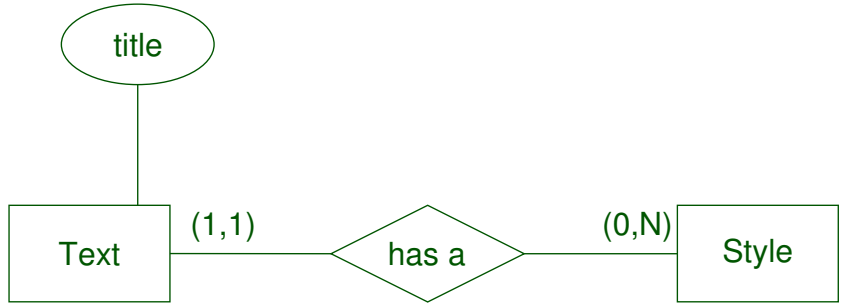


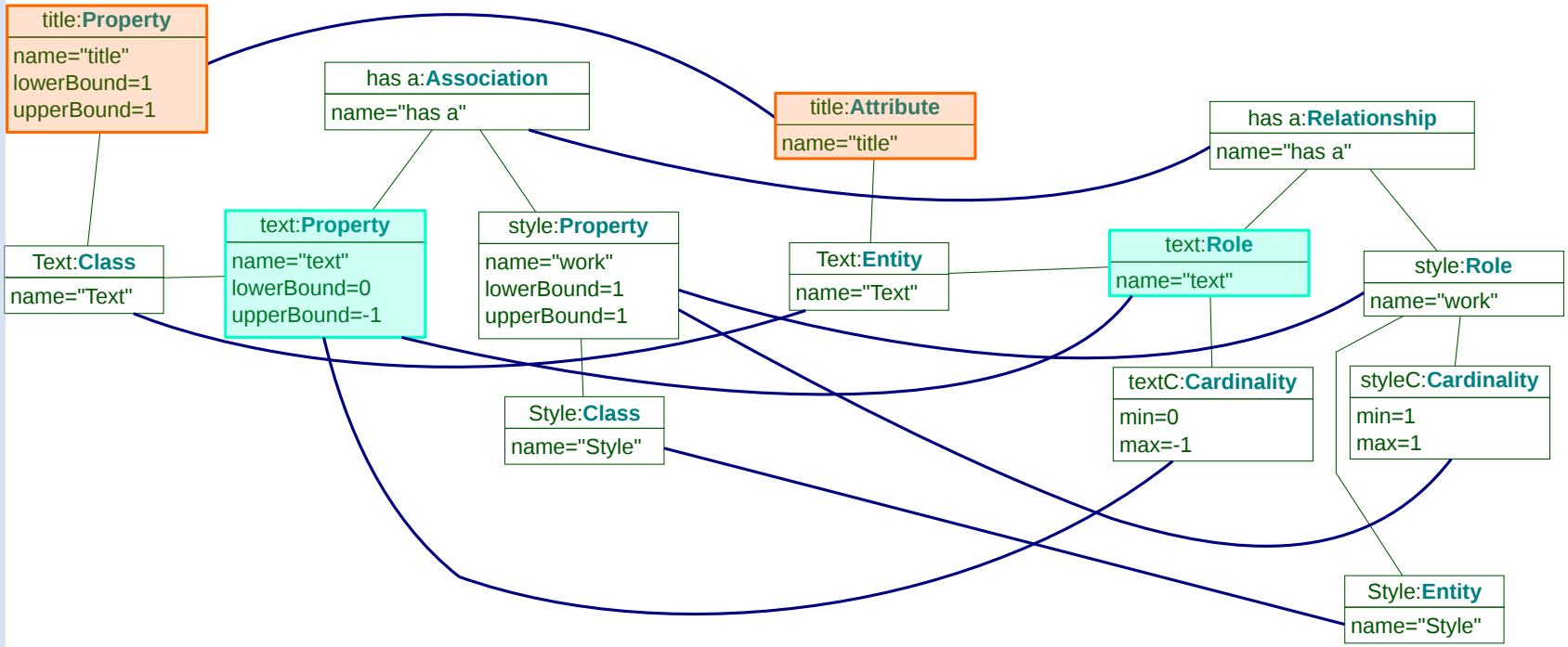
UML metamodel

Entity-Relationship metamodel

Input Data (excerpt)







Approach

- Consider the properties of the model elements:
  - their class
  - their relations with their neighbors
  - the properties of their neighbors
- Classify the different properties from the examples
- Classify the matching links considering the classification of the properties of their extremities

Formal Concept Analysis allows the classification of a set of objects considering their attributes

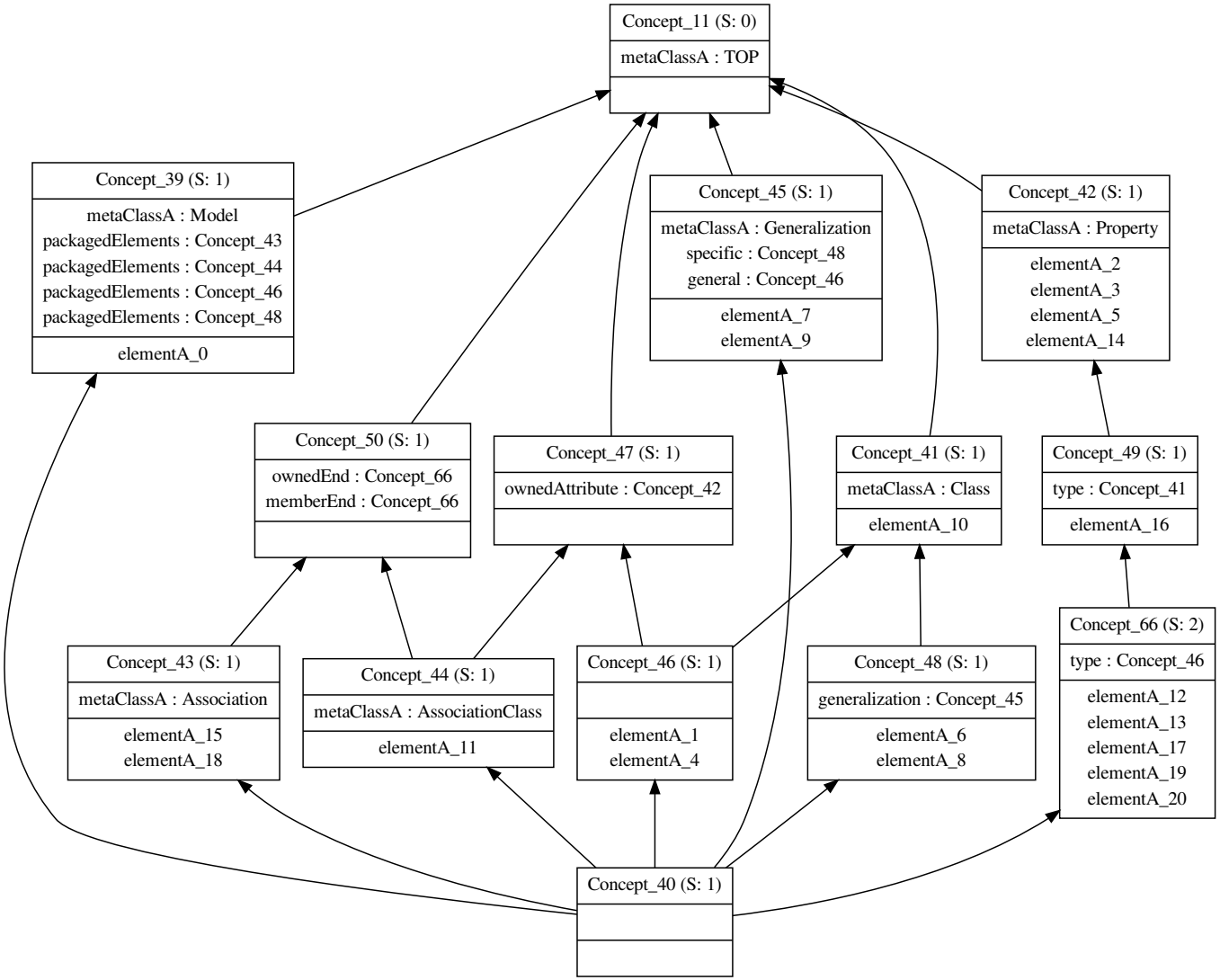
Transformation of models in tables

- Metamodel elements contexts
  - Source metamodel context
  - Target metamodel context
- Model elements contexts
  - Source model context
  - Target model context
- Matching links context
- Relations
  - between model source elements and their class from the source metamodel
  - between model target elements and their class from the target metamodel
  - between model source elements: *e.g. ownedAttribute*
  - between model target elements: *e.g. attribute*
  - between matching links and their source from the source model
  - between matching links and their target from the target model

The management of relations requires RCA to iterate

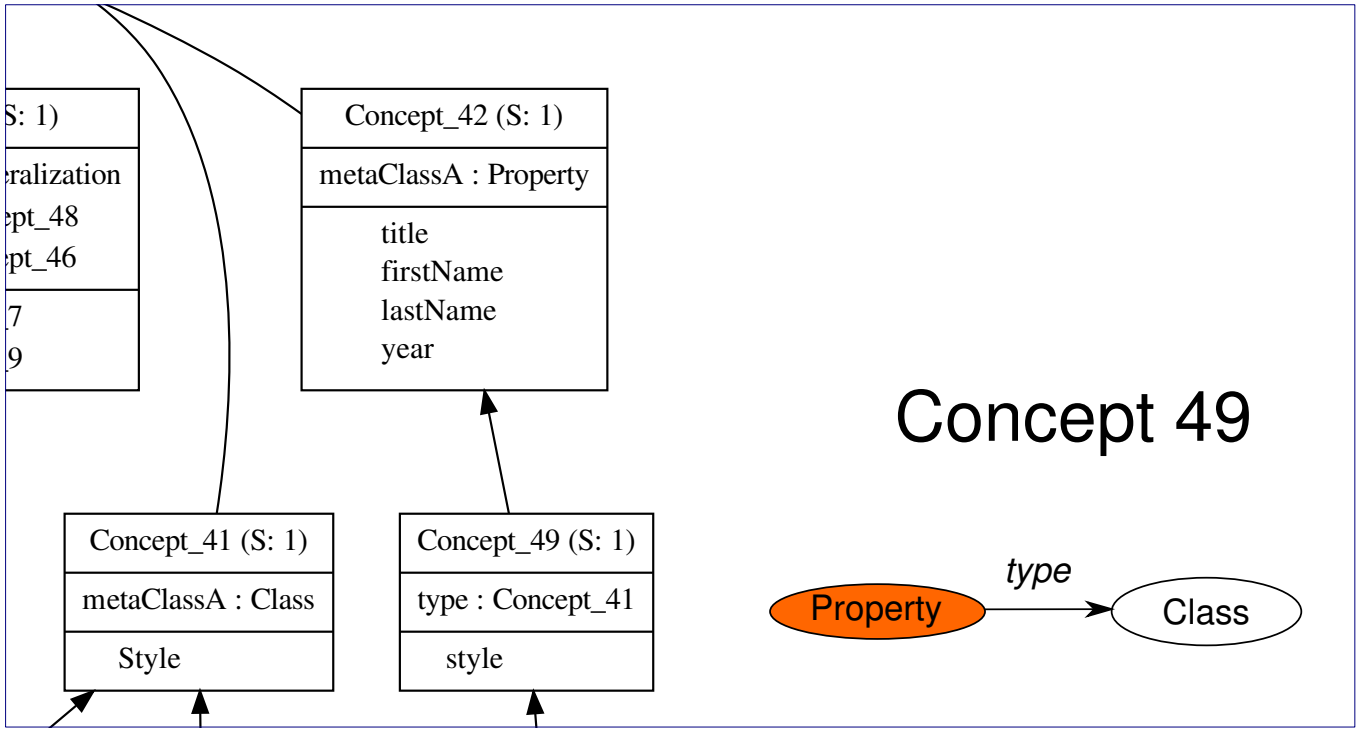
Lattice of model elements

Source model (UML)



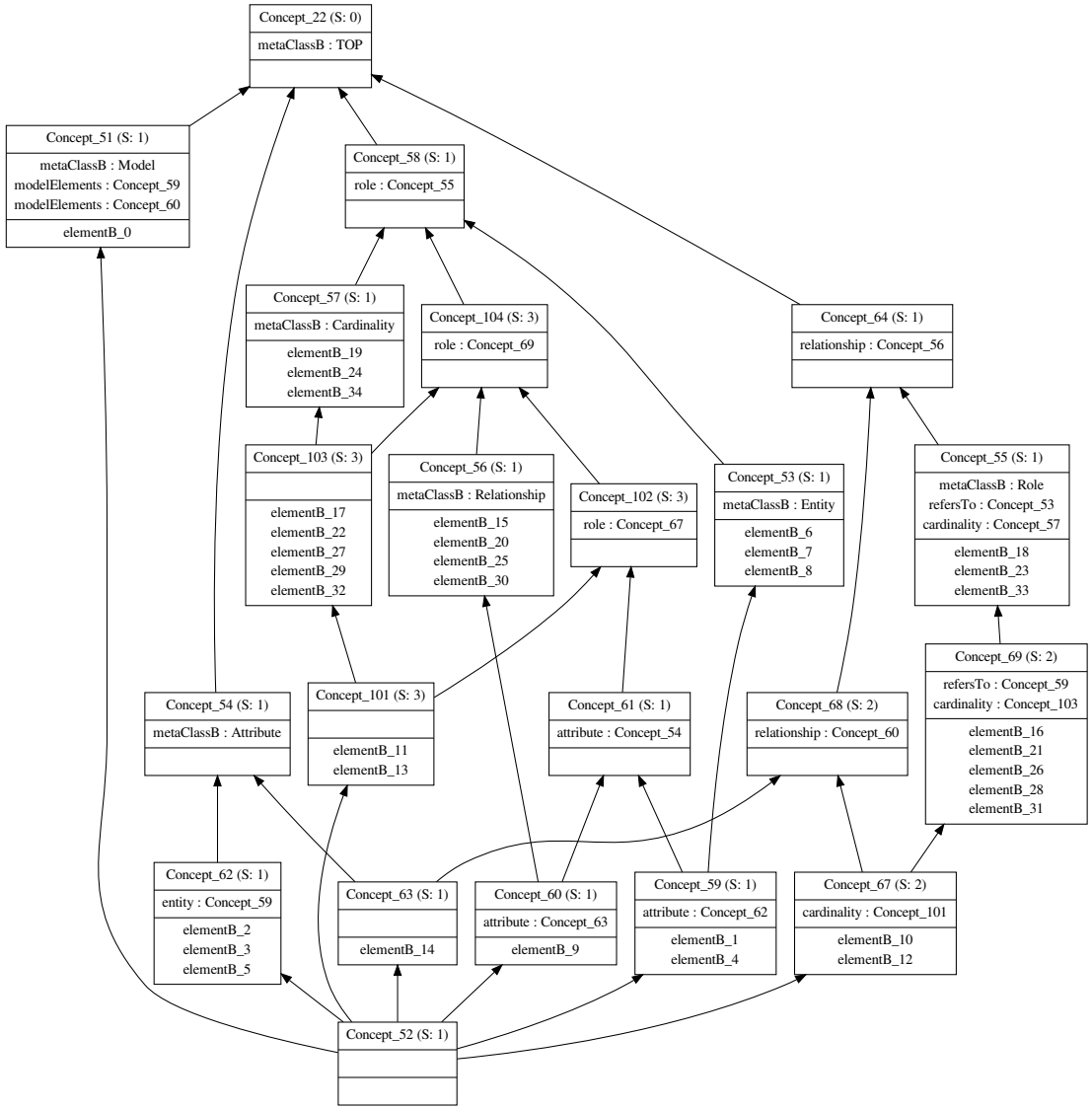
Lattice of model elements

Source model (UML)

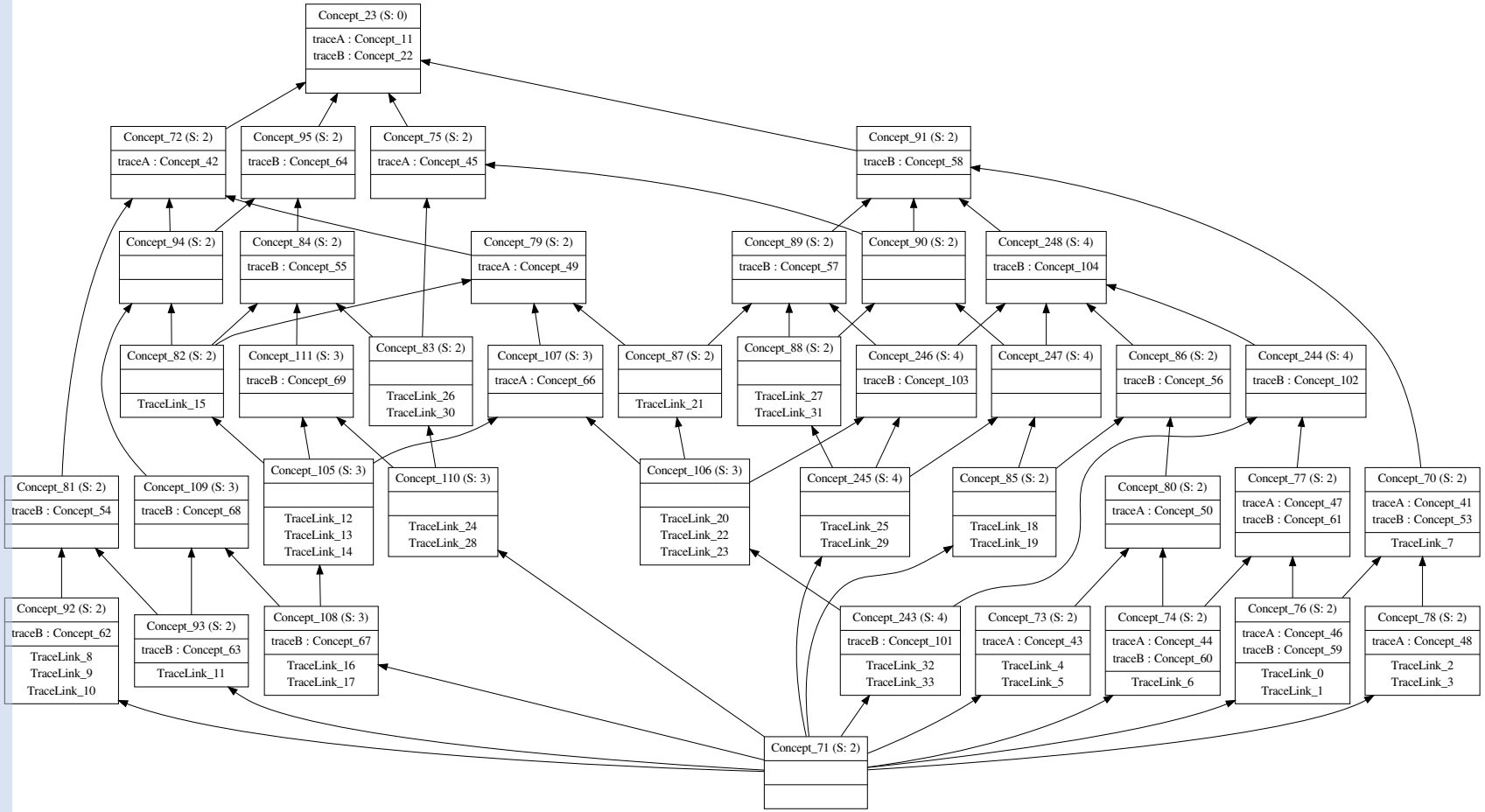


Lattice of model element

Target model (Entity-Relationship)

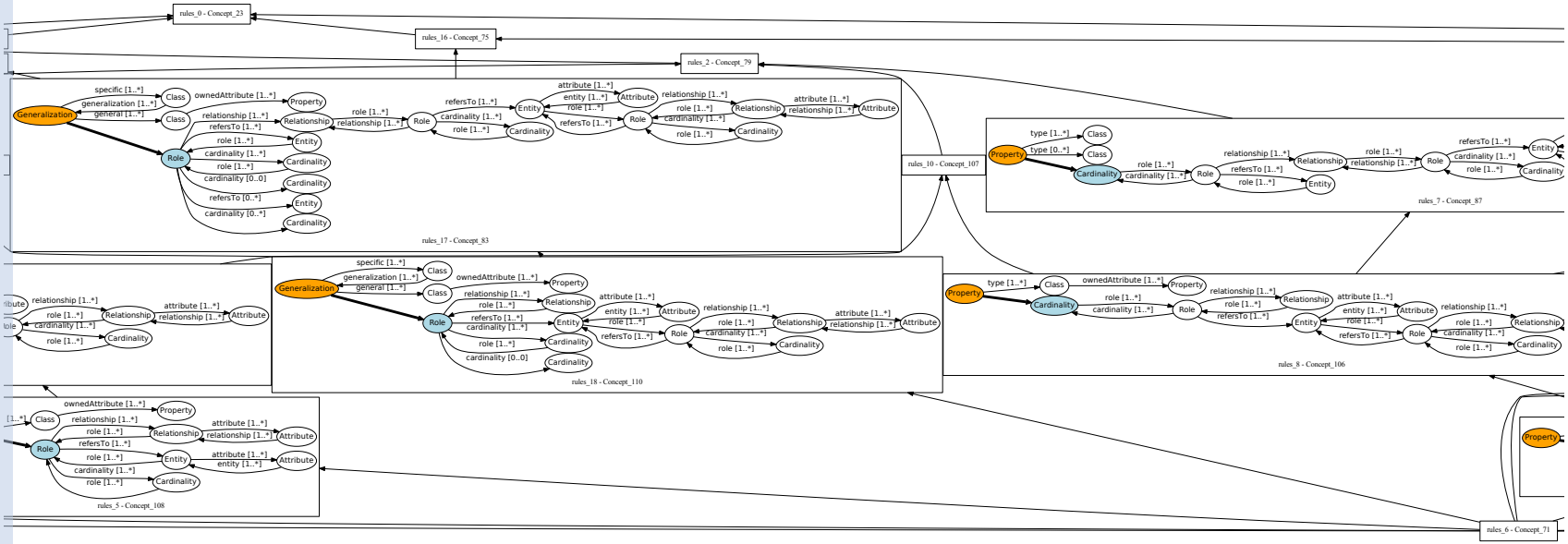


Lattice of matching links

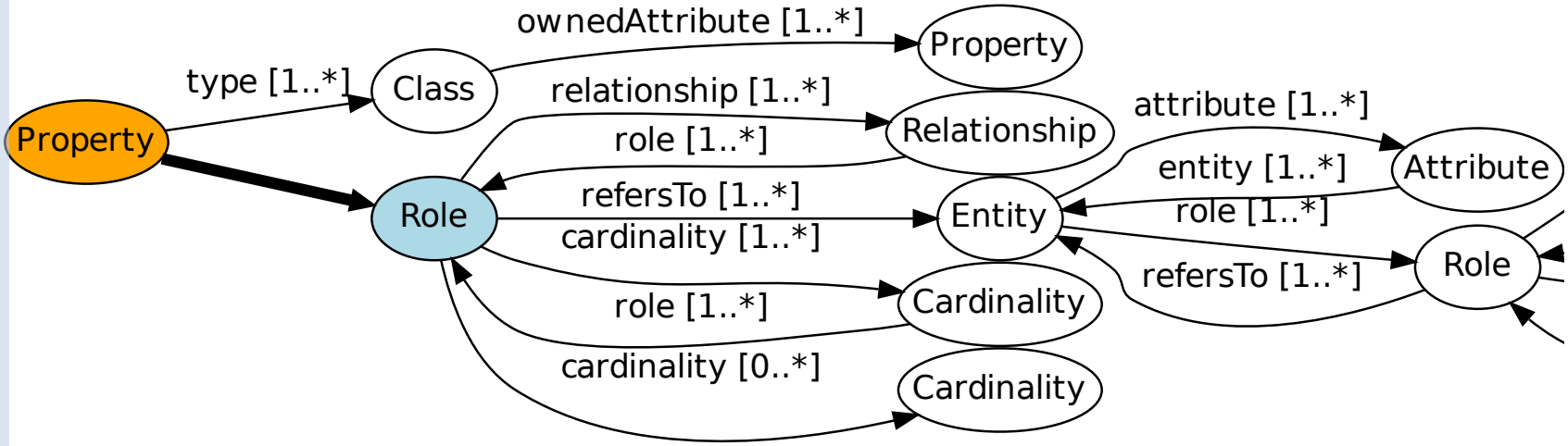


Lattice interpretation

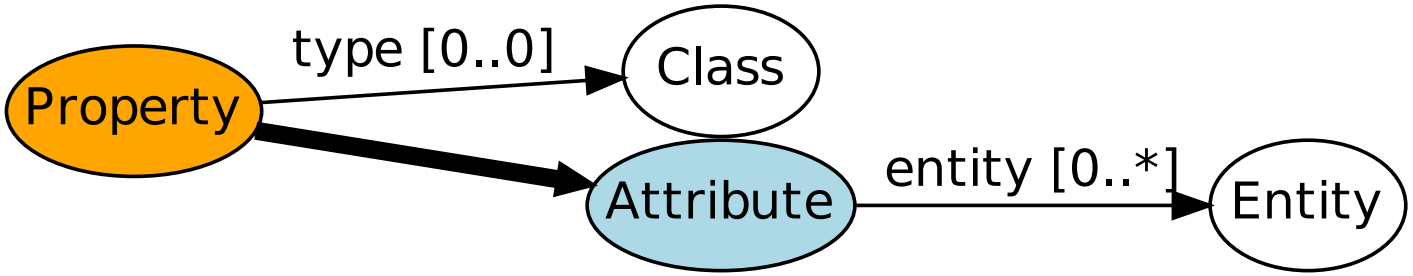
rules lattice (excerpt)



Rules examples



rules_4 - Concept_105



rules_11 - Concept_81

Lattice interpretation

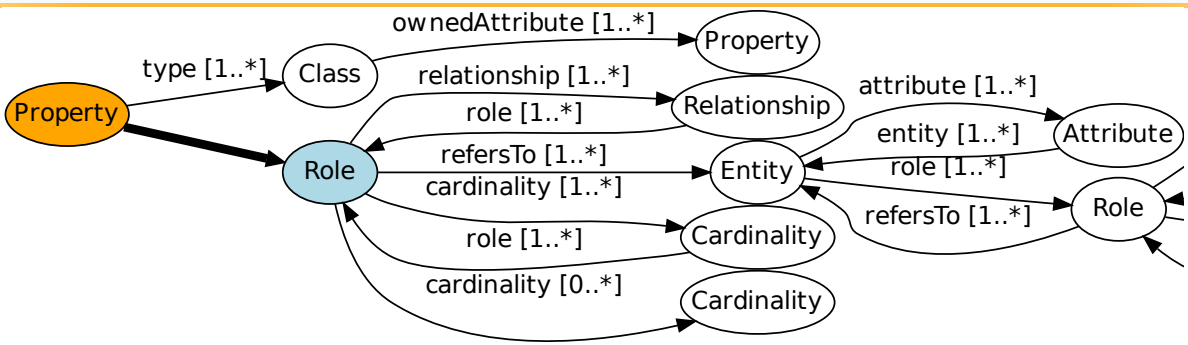
Hypothesis

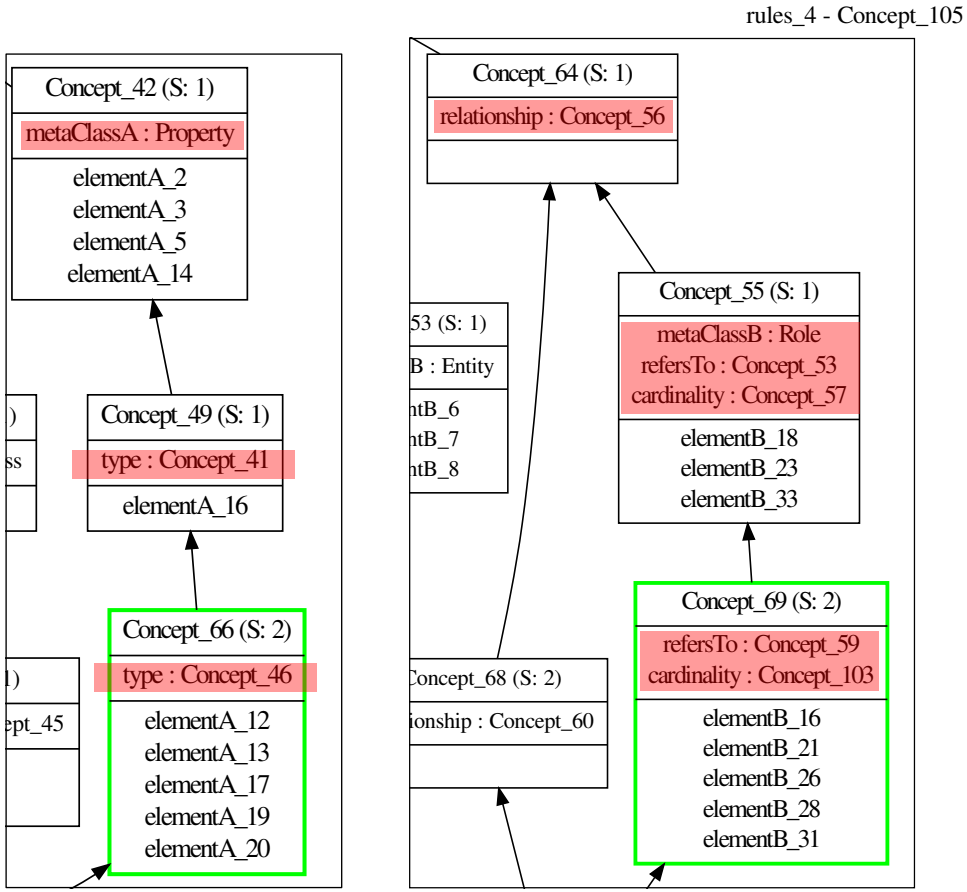
- closed world
- needs a good cover of the transformation

Different kinds of characteristics

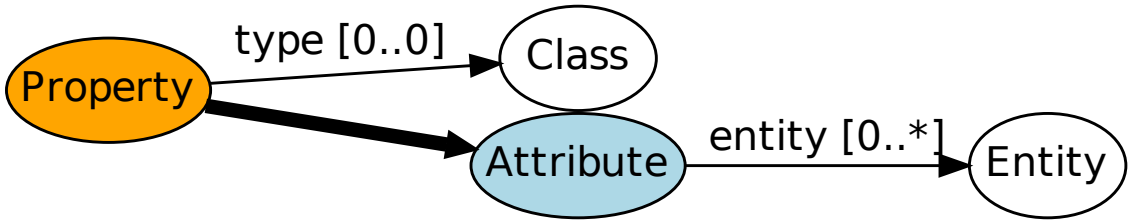
- needed characteristics
- allowed characteristics
- forbidden characteristics

Needed characteristics

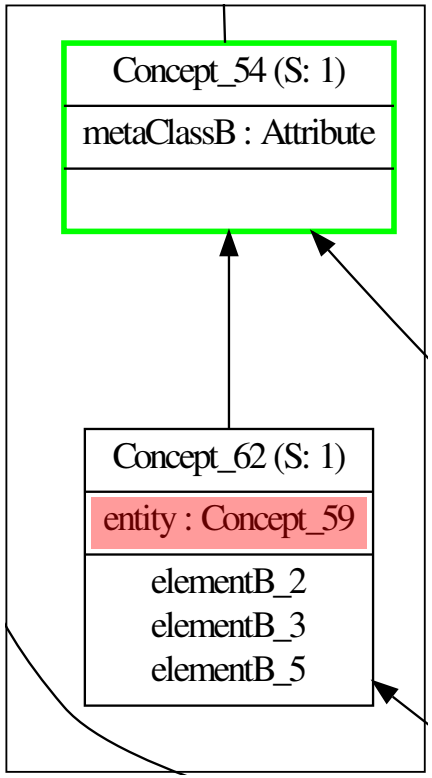




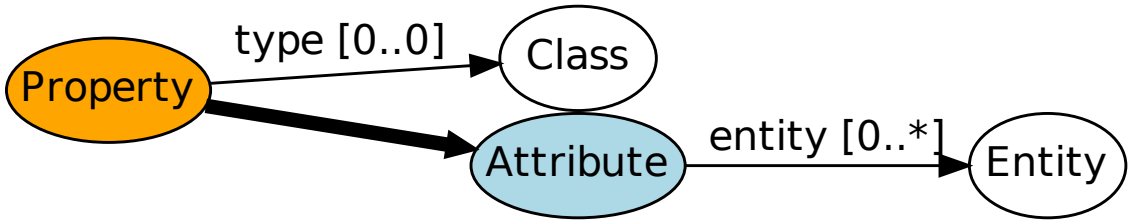
Allowed characteristics



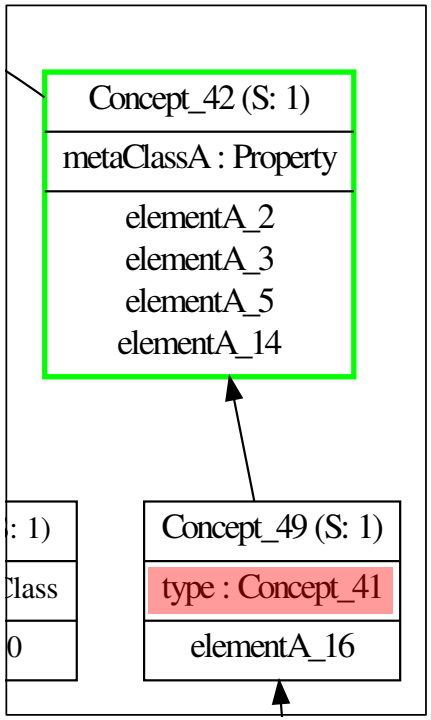
rules_11 - Concept_81



Forbidden characteristics



rules_11 - Concept_81



Empirical study

name	size of MapLinks lattice	number of extracted rules	premise size				premise depth			
			min	max	avg.	std. dev.	min	max	avg.	std. dev.
associations-persons	9	6	1	5	4	1.53	1	3	5	0.82
Book2Publication	4	2	2	2	2	0	2	2	4	1.21
delegation1	47	12	15	33	18	5.89	4	8	5	1.22
delegation2	8	5	7	17	9	4	3	5	3	0.87
Disaggregation	22	10	6	10	6	1.26	4	6	5	0.8
Ecore2Class	14	8	7	15	8	3.16	3	6	5	0.95
EliminateRedundantInheritance	8	6	7	11	8	1.63	4	6	4	1.27
emf2km3	26	19	14	52	20	9.48	5	7	4	0.77
EquivalenceAttributesAssociations	47	18	11	36	15	7.96	3	6	4	0.77
extractClass	10	4	6	6	6	0	3	4	4	1.15
Families2Persons_final	11	8	1	7	5	1.8	1	3	4	1.22
hideDelegate	53	22	17	117	35	28.7	4	7	2	0
IntroducePrimaryKey	31	17	11	18	12	2.5	4	7	2	0.91
IntroducingInterface	38	11	10	20	10	3.02	3	6	4	1.61
JavaSource2Table	17	10	5	9	7	1.3	3	7	2	0.87
KM32EMF	18	15	11	20	14	3.74	4	6	3	1.35
KM32Problem	37	28	28	378	52	65.36	7	9	8	0.65
uml-er	35	24	2	11	6	2.75	2	9	5	1.15
uml-er-iccs	21	12	3	9	6	1.78	2	5	7	1.54
uml-er2	25	15	2	9	4	2.19	2	6	3	1.41
uml-er3	40	19	2	4	3	0.83	2	4	2	0.97
UML2ClassDiagram_to_KM3	200	40	28	187	58	40.51	5	10	5	2.72

Empirical study

nom	conclusion size				conclusion depth			
	min	max	avg.	std. dev.	min	max	avg.	std. dev.
associations-persons	1	4	2	0.91	1	3	2	0.58
Book2Publication	1	1	1	0	1	1	1	0
delegation1	7	17	7	2.89	3	6	4	1.04
delegation2	15	26	20	4.96	5	6	5	0.63
Disaggregation	7	11	7	1.26	4	6	4	0.89
Ecore2Class	1	8	4	2.26	1	4	3	1
EliminateRedundantInheritance	7	11	8	1.63	4	6	5	0.82
emf2km3	11	20	14	3.59	4	6	4	1.17
EquivalenceAttributesAssociations	9	38	14	8.37	4	7	5	1.2
extractClass	16	20	18	2	5	6	5	0.87
Families2Persons_final	1	4	2	0.79	1	3	2	0.5
hideDelegate	17	186	39	37.75	4	7	5	0.88
IntroducePrimaryKey	9	14	9	1.21	4	6	4	1.03
IntroducingInterface	2	20	9	4.53	2	6	4	1.48
JavaSource2Table	1	2	1	0.63	1	2	1	0.63
KM32EMF	14	52	21	10.23	4	7	5	1.1
KM32Problem	1	1	1	0	1	1	1	0
uml-er	2	15	9	3.61	2	8	5	1.88
uml-er-iccs	5	6	5	0.29	3	4	3	0.76
uml-er2	2	15	9	3.71	2	8	5	1.88
uml-er3	2	15	9	3.47	2	8	5	1.84
UML2ClassDiagram_to_KM3	18	45	22	6.06	4	7	5	0.74

Synthesis

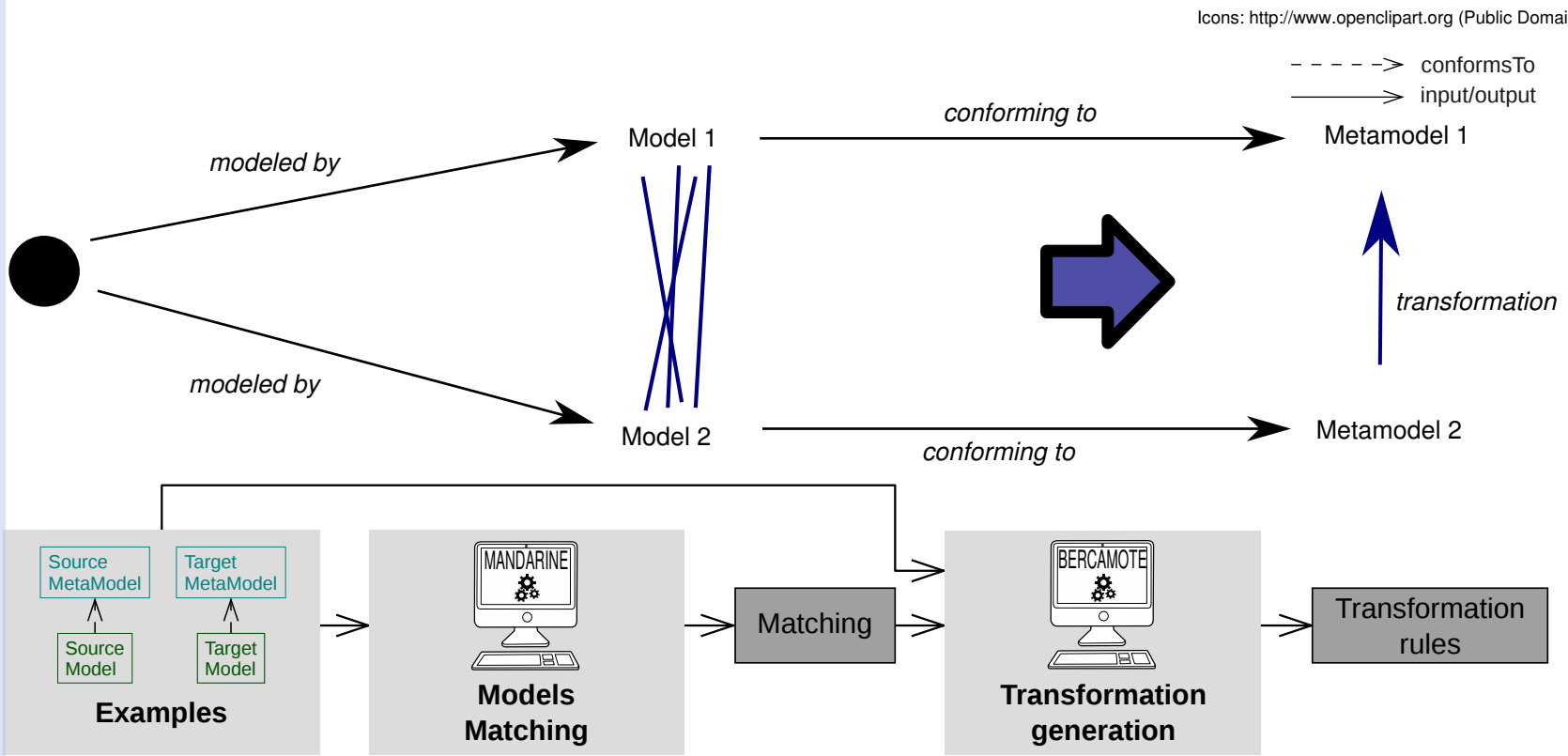
The generation of transformation patterns is the result of the following steps

- the transformation of the examples into relational contexts
- the application of the process RCA
- the interpretation of the final lattices

Outline

Contribution

An approach with tool for assisting the development of model transformations



Contribution

- Implementation independent with transformation metamodels
- Model matching assisted
- Result usable thanks to lattices
- Result genericity

Perspectives

Model matching

- Improving precision on the AnchorPROMPT adaptation
- Adapt other matching approaches for model matching

Transformation rules generation

- Building relations in final model for not simple case
- Improving interaction with user
- Insisting on approach validation

Evolutions

- Adapt the approach to other problems
- Investigating complementarity with other approaches

Thank you for your attention

Metamodel matching

Lopes et al.(2005)
Generating transformation definition from mapping specification: Application to web service platform.
In CAiSE'05, LNCS 3520, pages 309–325, 2005.

Del Fabro et Valduriez(2007)
Semi-automatic model integration using matching transformation and weaving models.
In International Conference SAC'07, pages 963–970. ACM, 2007.

Falleri et al.(2008)
Meta-model Matching for Automatic Model Transformation Generation.
In MODELS'08, LNCS 5301, pages 326–340. Springer, 2008.

Model transformation by example

Balogh et Varró(2009)
Model transformation by example using inductive logic programming.

Software and Systems Modeling, 8(3):347–364, 2009.

Kessentini et al.(2008)
Model Transformation as an Optimization Problem.
In MODELS'08, LNCS 5301, pages 159–173. Springer, 2008.

Wimmer et al.(2007)
Towards model transformation generation by-example.
In HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences, page 285b, Washington, DC, USA, 2007. IEEE Computer Society.