UNIVERSITÄT KOBLENZ · LANDAU

IWVI

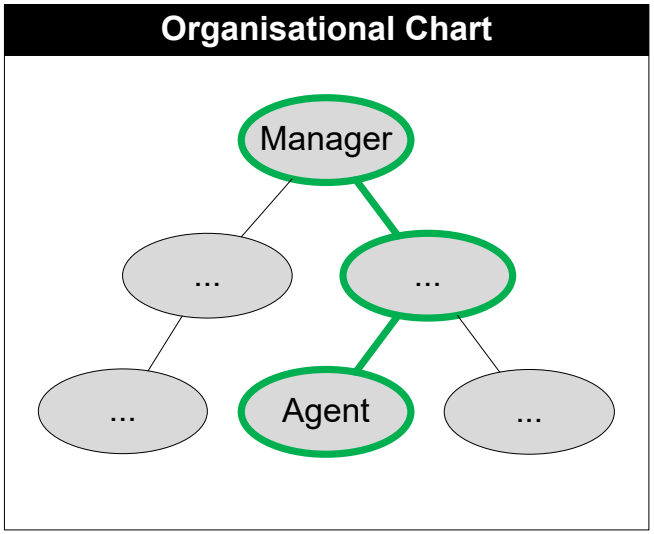
BUSINESS PROCESS MANAGEMENT

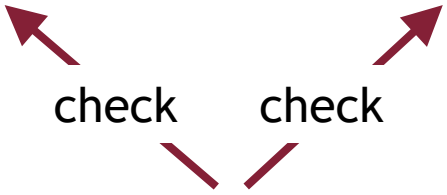
MODEL QUERY III:
THE DIAGRAMED MODEL QUERY LANGUAGE (GRAPH MATCHING)

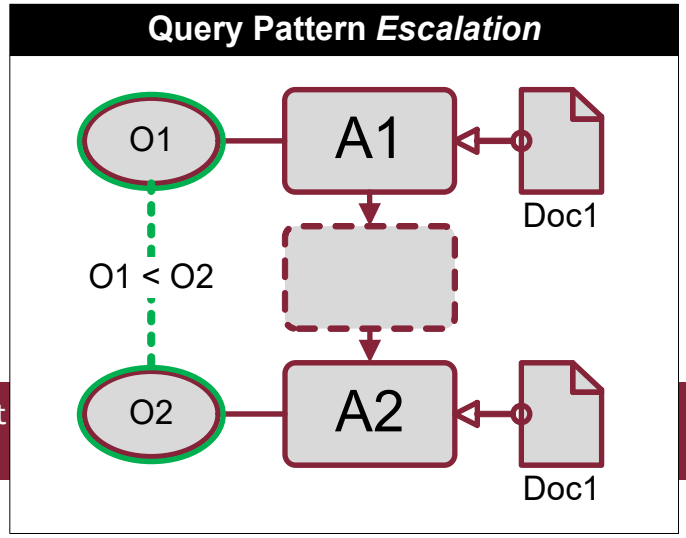
AGENDA

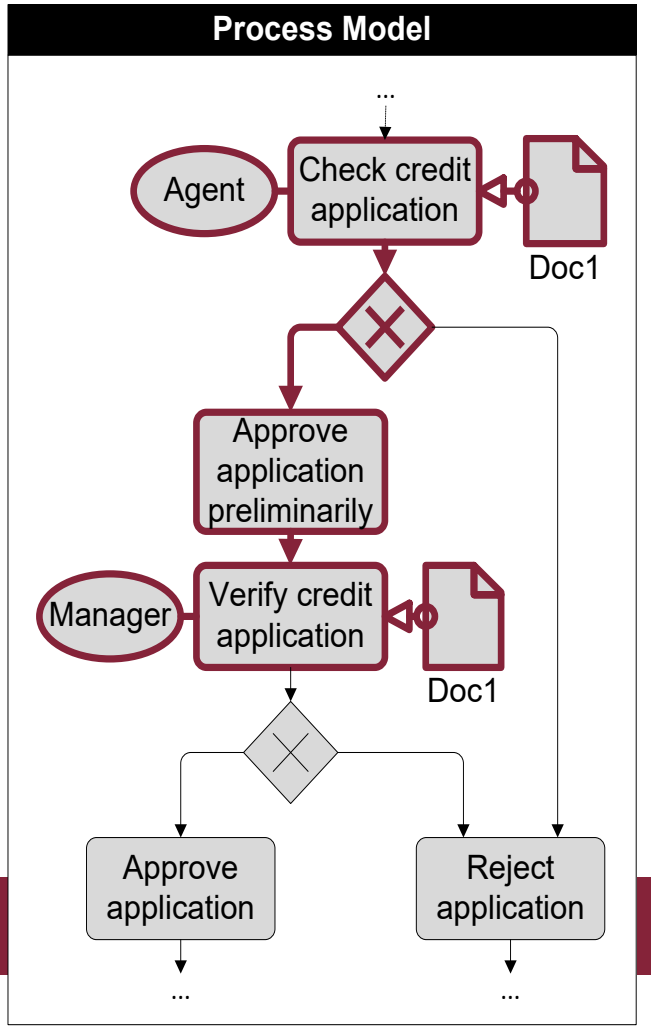
- Motivation for a Diagramed Model Query Language (DMQL)
- Requirements of DMQL
- The Diagramed Model Query Language (DMQL) incl. Parallel Tool Presentation
- Further Information on the Topic Model Query

MOTIVATING EXAMPLE: BUSINESS PROCESS COMPLIANCE MANAGEMENT



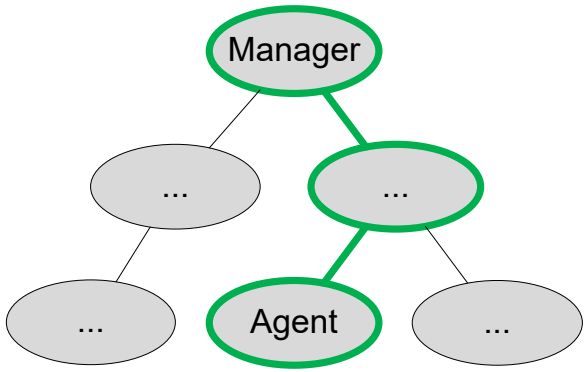




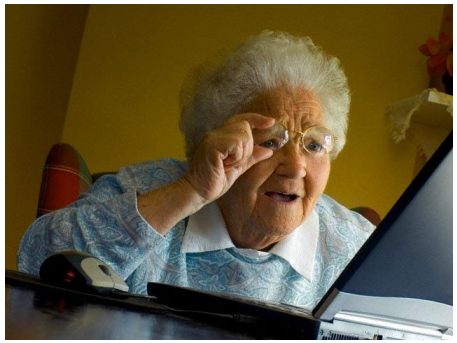


GMQL (AND RELATED GRAMMAR-BASED QUERY LANGUAGES)

Organisational Chart



check check

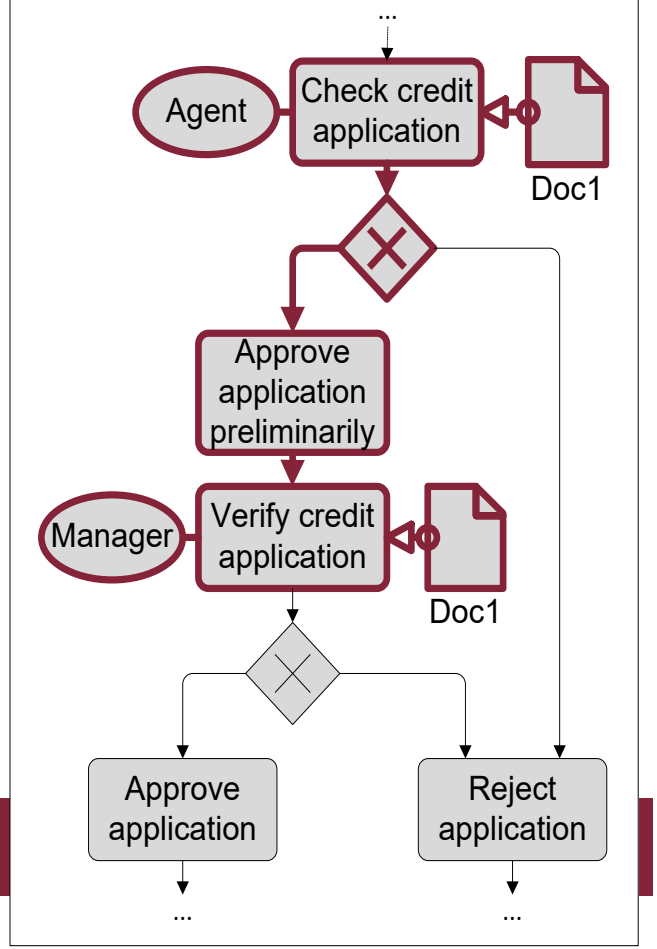


Query Pattern *Escalation*

```
DP(
    SELF_UNION( INNER_INTERSECTION(
        EDR(
            EOT(O,Function),
            EWAOV(
                EOT(O,OrgUnit),
                Caption,
                A
            )
        ),
        DS(
            EWAOV(
                EOT(O,Data),
                Caption,
                C
            ),
            EOT(O,Function)
        ),
    )),
    SELF_UNION( INNER_INTERSECTION(
```

?

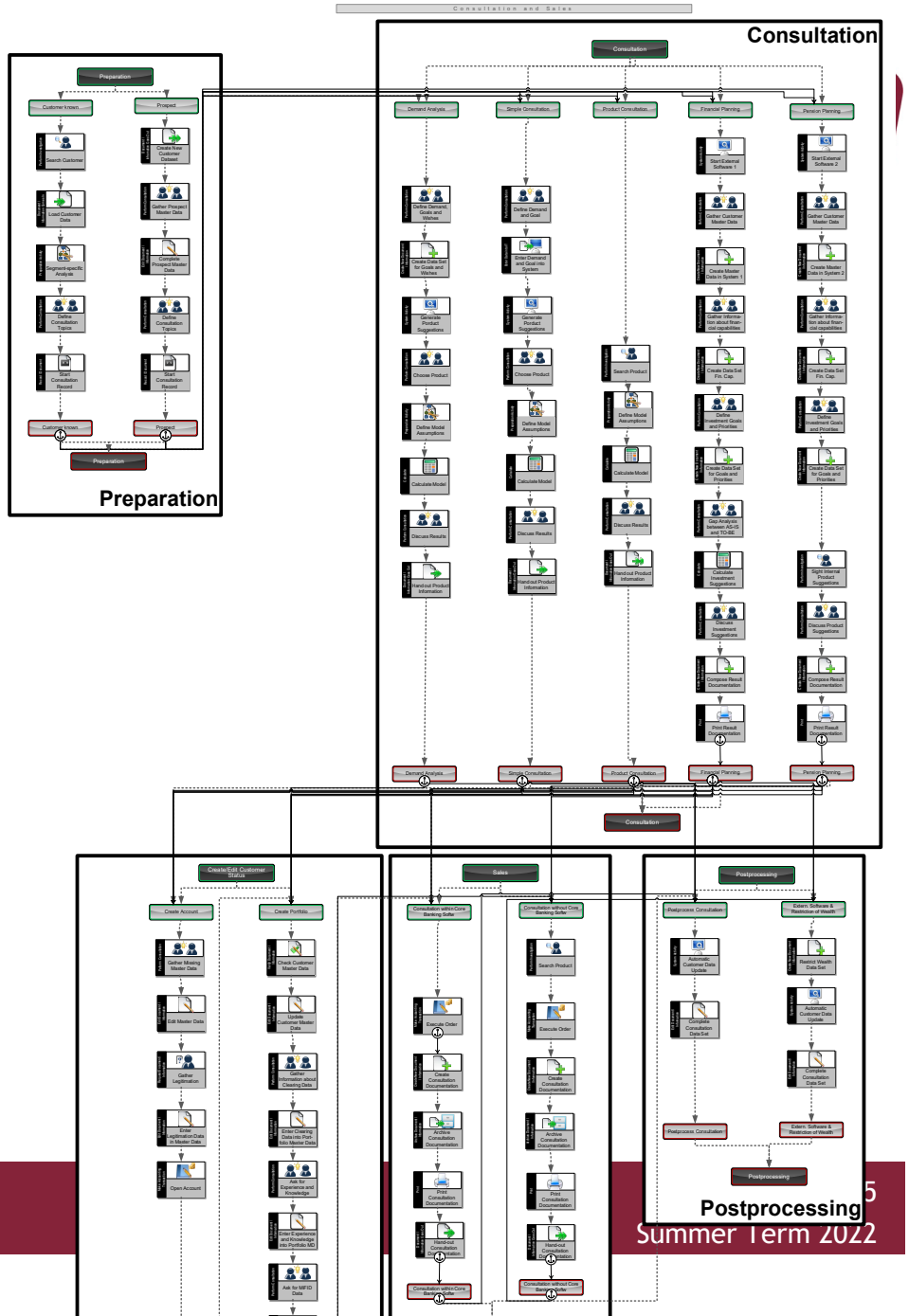
Process Model



EVALUATION

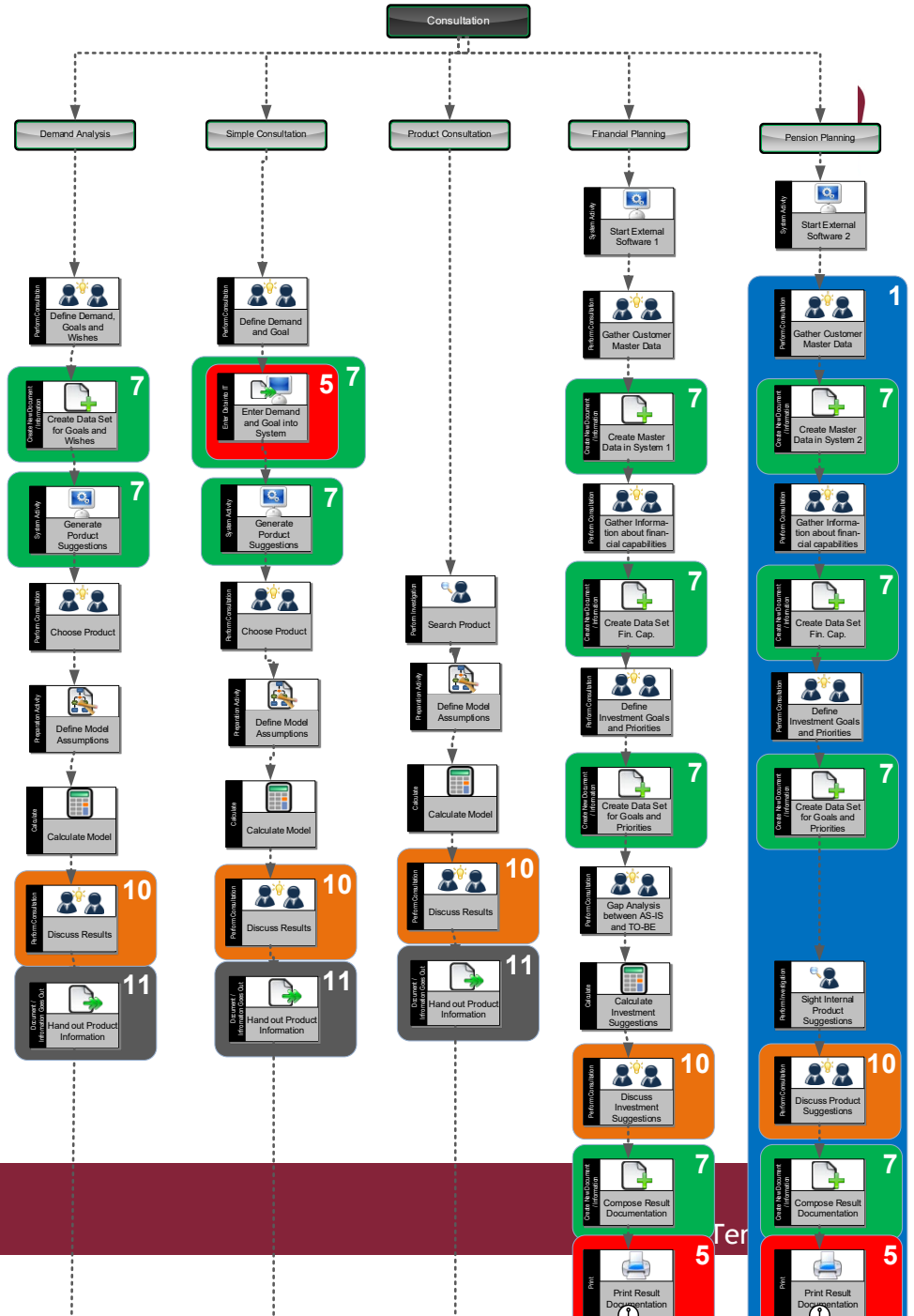
- Business processes of an IT service provider for banks
- Banking software
- PICTURE models

- 13 exemplary compliance rules from WpHG, InvG, WpDVerOV, KWG, GWG



EVALUATION

- Several matches
- Potential compliance violations
- Several matches were actual compliance violations
- Impact: adaptations of business software for several 1000s of banks



EVALUATION

No.	Factor	Focus Group	Very Low — Results — Very High
1.	**Ease of Use** „How do you estimate the effort necessary to use the approach for process compliance checking in your organization?“	FG1	0 — 0.43 — 0.5 — 0.54 — 1
		FG2	0 — 0.5 — 0.6 — 0. — 1
2.	**Job relevance** „How do you estimate the relevance of the functionality provided by the approach for your job?“	FG1	0 — 0.77 — 0.82 — 0.86 — 1
		FG2	0 — 0.25 — 0.5 — 0.75 — 1
3.	**Output quality** „How do you estimate the quality of pattern search results?“	FG1	0 — 0.75 — 0.81 — 0.87 — 1
		FG2	0 — 0.75 — 0.76 — 0.80 — 1
4.	**Perceived usefulness** „How do you perceive the potential benefit of the approach for your job?“	FG1	0 — 0.48 — 0.63 — 0.7 — 1
		FG2	0 — 0.55 — 0.66 — 0.77 — 1
5.	**Behavioral intention** „How do you estimate the probability of using the approach?“	FG1	0 — 0.54 — 0.61 — 0.74 — 1
		FG2	0 — 0.39 — 0.75 — 0.79 — 1
6.	**Frequency of usage** „How often would you use the approach?“ (scale from very seldom to very often)	FG1	0 — 1
		FG2	0 — 0.46 — 0.89 — 0.92 — 1

Caption

1. Quartile (Q_1) Median 3. Quartile (Q_3)

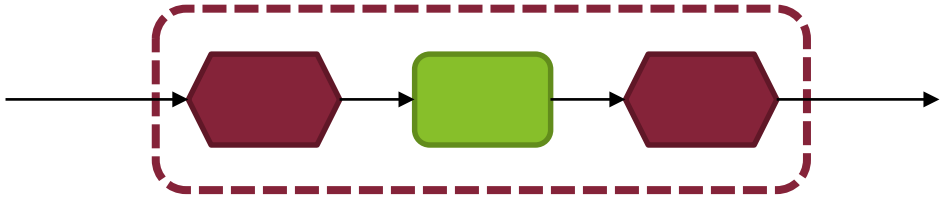
AGENDA

- Motivation for a Diagramed Model Query Language (DMQL)
- Requirements of DMQL
- The Diagramed Model Query Language (DMQL) incl. Parallel Tool Presentation
- Further Information on the Topic Model Query

THE DIAGRAMED MODEL QUERY LANGUAGE (DMQL)

- Requirements: see GMQL
- Plus:
  - graphical specification of patterns
  - check different kinds of models simultaneously (cf. example)
- A formalization of the graphic pattern is used as input for a graph pattern matching algorithm

DMQL pattern



what‘s found in an EPC, e.g.

THE DIAGRAMED MODEL QUERY LANGUAGE (DMQL)

- Requirements: see GMQL
- Plus:
  - graphical specification of patterns
  - check different kinds of models simultaneously (cf. example)
- A formalization of the graphic pattern is used as input for a graph pattern matching algorithm

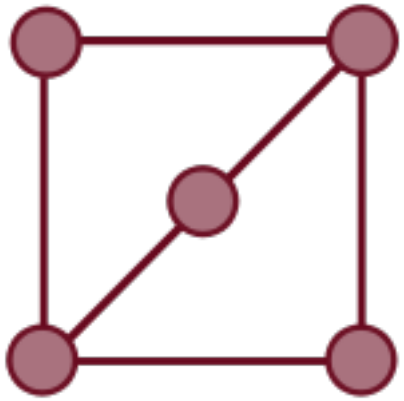
- Graph pattern matching incl.
  - isomorphic parts (edge-edge-mapping)
  - homeomorphic parts (edge-path-mapping)
  - vertex and edge attributes
  - logical combination of pattern elements

THE DIAGRAMED MODEL QUERY LANGUAGE (DMQL)

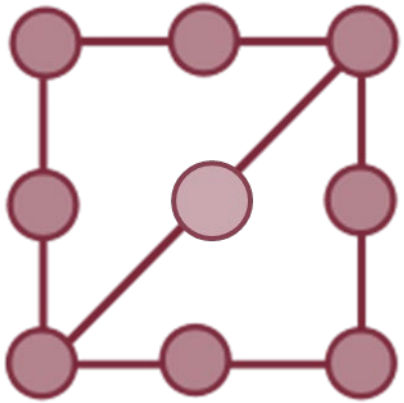
- HomEomorphism: A graph G and G' are homeomorphic, if there is a graph isomorphism from a subdivision of G, to a subdivision of G'

THE DIAGRAMED MODEL QUERY LANGUAGE (DMQL)

- HomEomorphism: A graph G and G' are homeomorphic, if there is a graph isomorphism from a subdivision of G, to a subdivision of G'



G



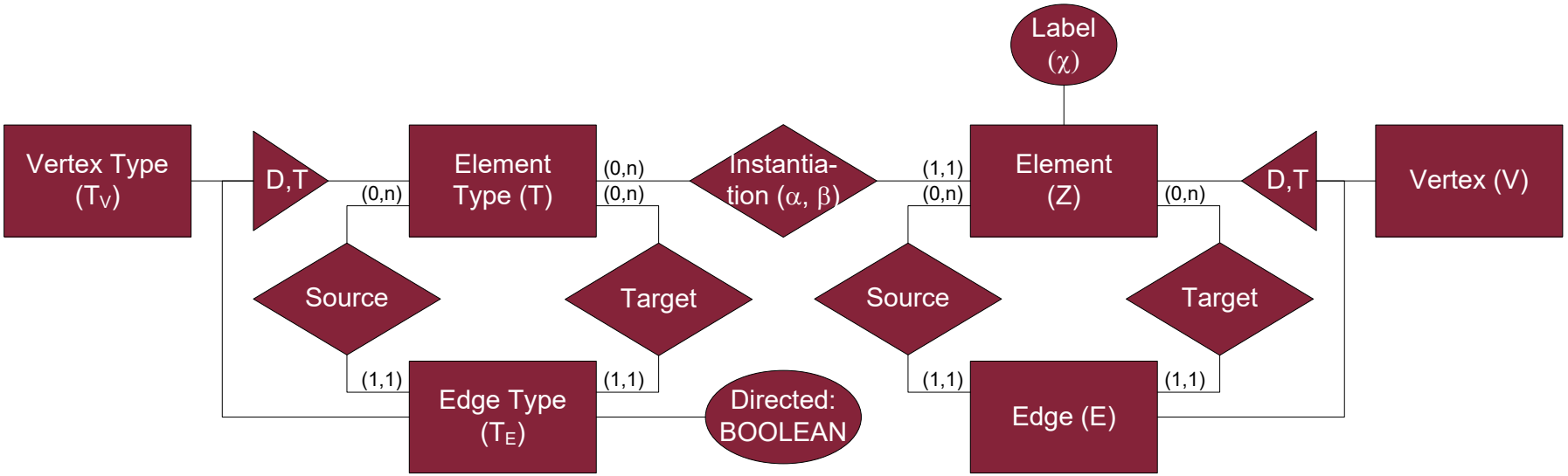
G'

AGENDA

- Motivation for a Diagramed Model Query Language (DMQL)
- Requirements of DMQL
- The Diagramed Model Query Language (DMQL) incl. Parallel Tool Presentation
- Further Information on the Topic Model Query

BASIC SPECIFICATIONS

MULTIPLE MODELING LANGUAGES AND VIEWS



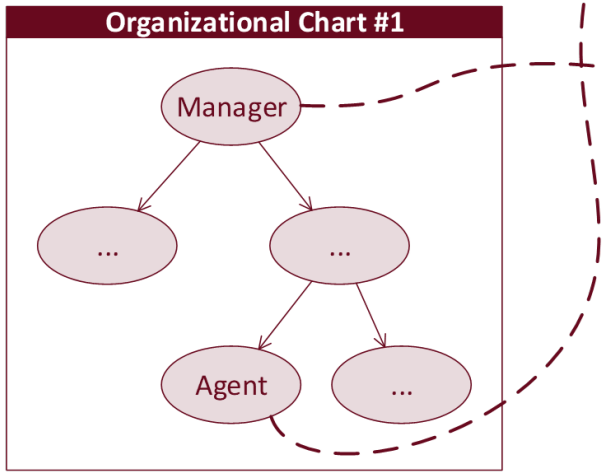
PRELIMINARIES: INTEGRATED CONCEPTUAL MODELS

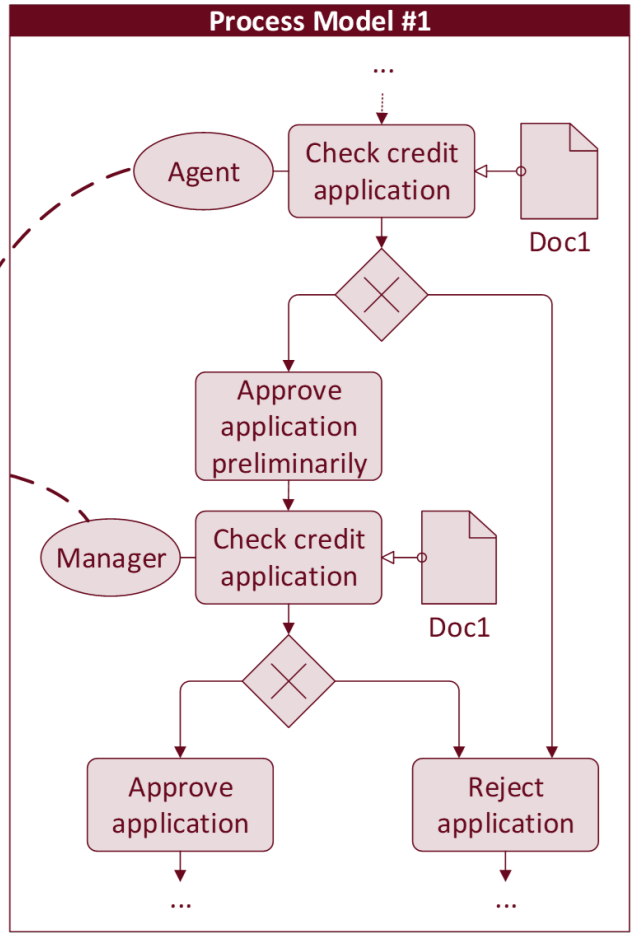
Vertex	Model
Agent	OrgChart
Manager	OrgChart
Agent	Process Model
Manager	Process Model

Model	Language
OrgChart #1	OrgChart
Process Model #1	Process Model

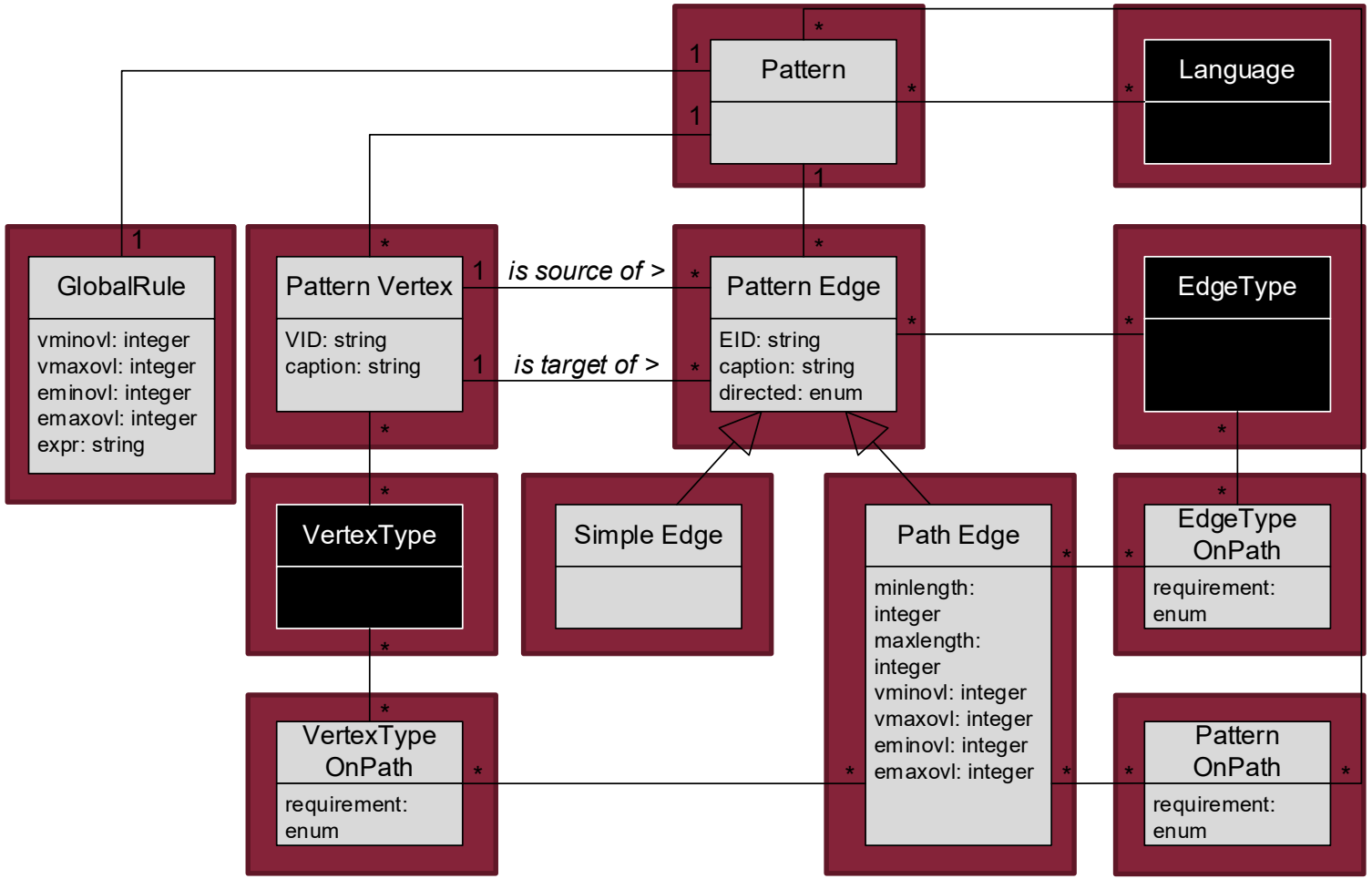
Vertex Type	Language
OrgUnit	OrgChart
OrgUnit	Process Model

Vertex	Vertex Type
Agent	OrgUnit
Manager	OrgUnit
…	…

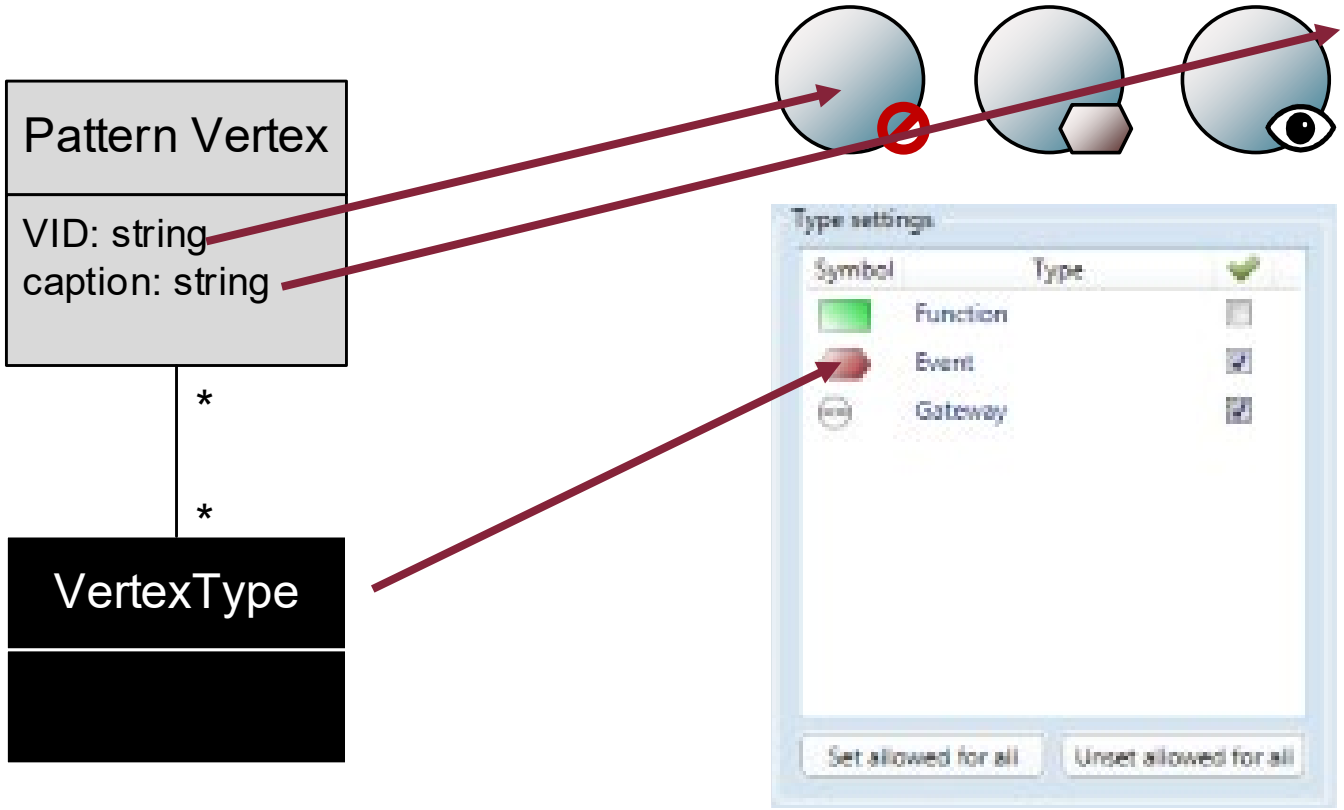




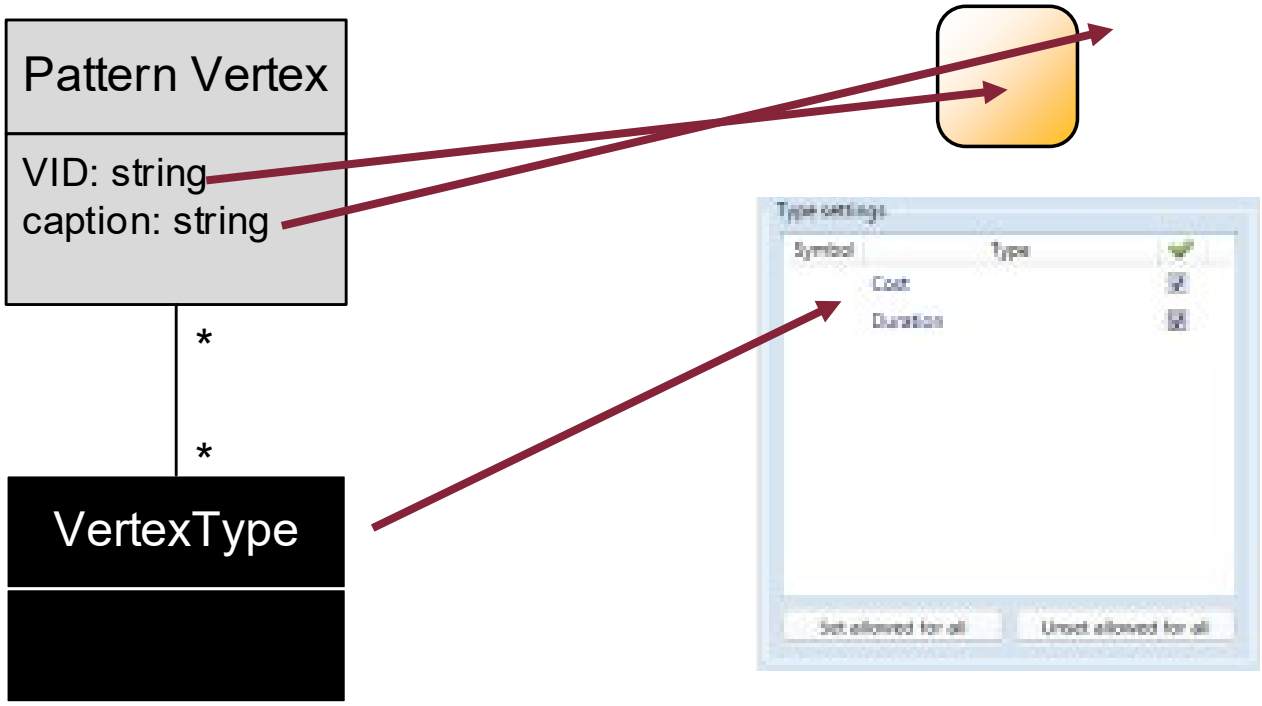
ABSTRACT SYNTAX



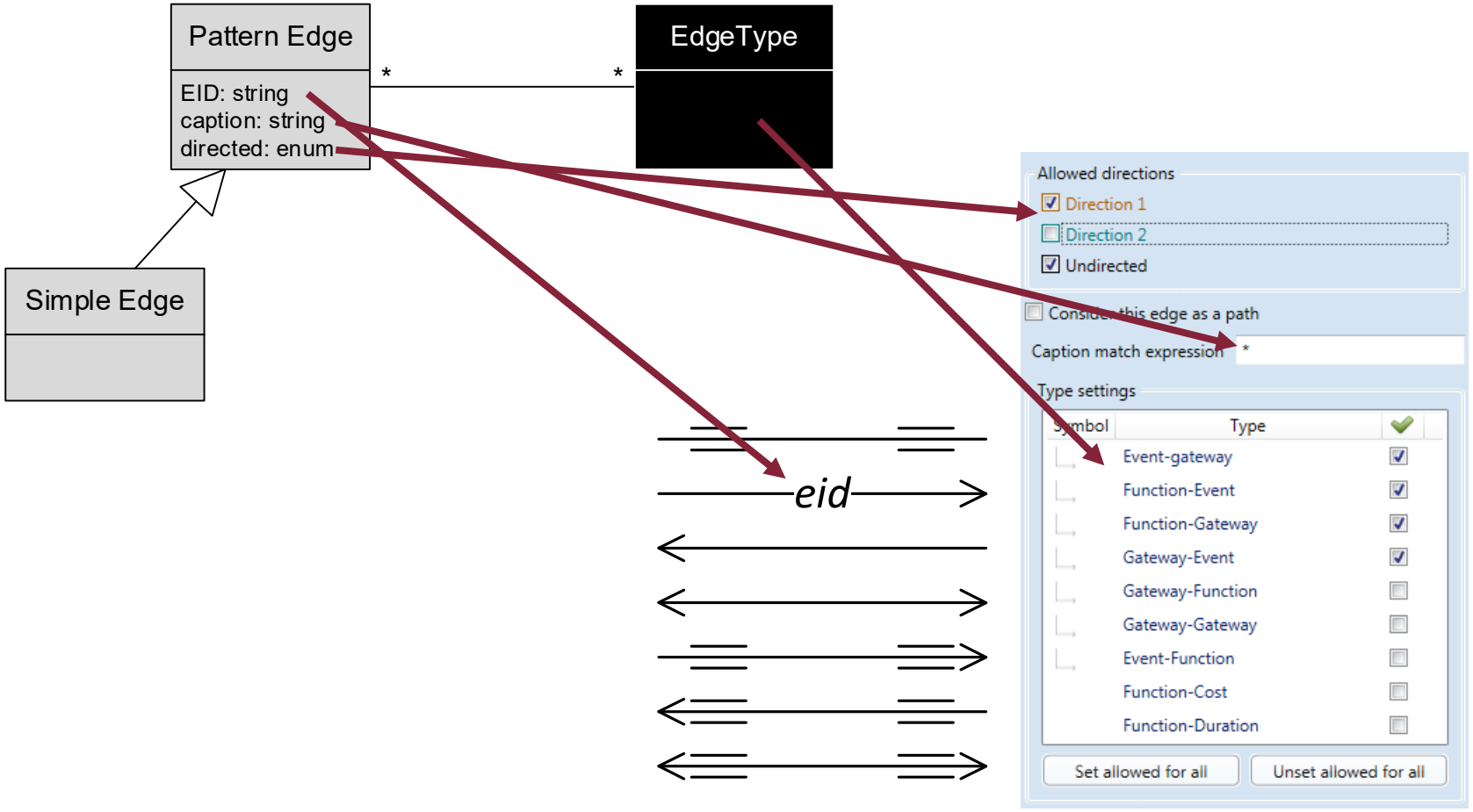
CONCRETE SYNTAX



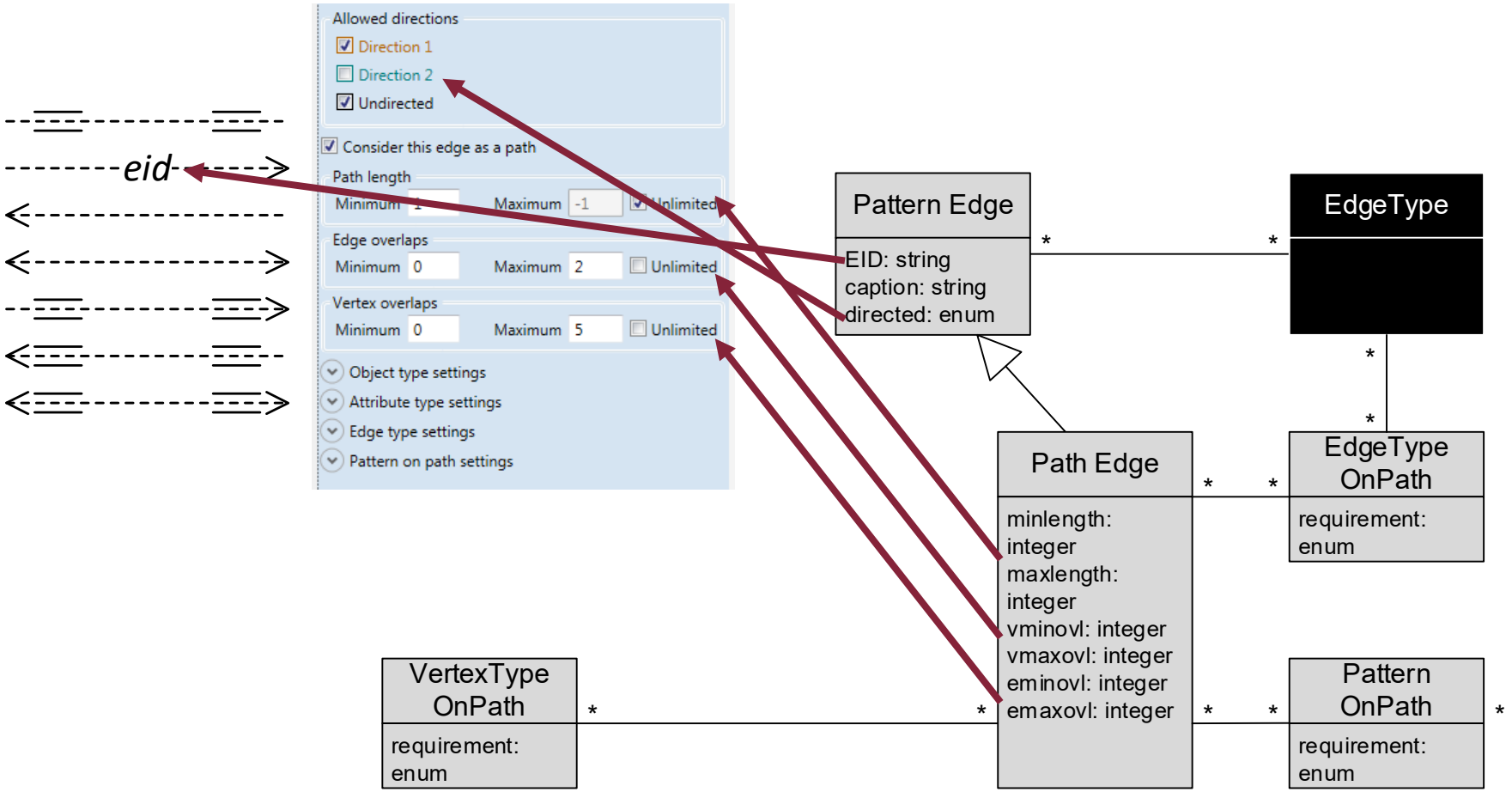
CONCRETE SYNTAX



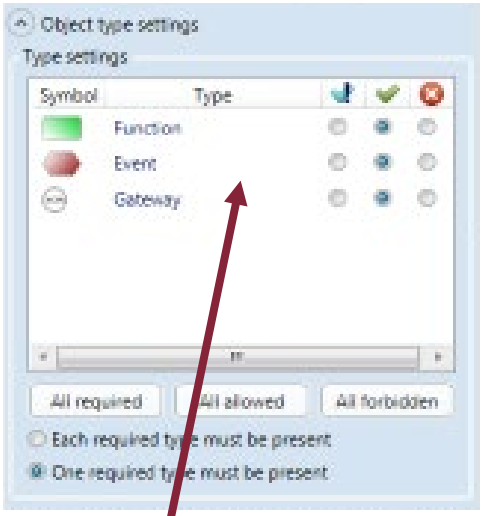
CONCRETE SYNTAX

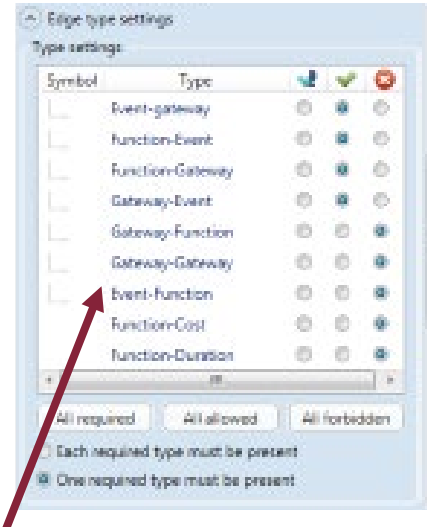


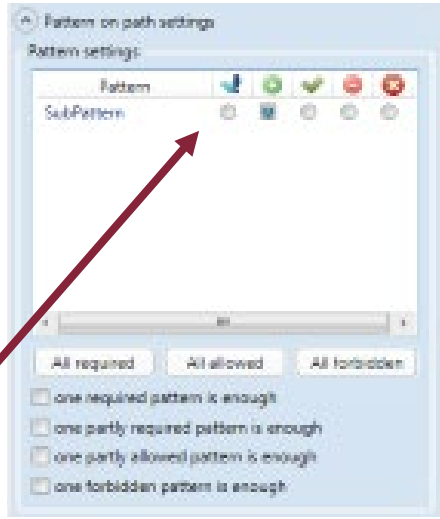
CONCRETE SYNTAX



CONCRETE SYNTAX







CONCRETE SYNTAX

GLOBAL RULES

- Global rules can be used to formulate constraints on patterns
- An exemplary Global Rule could look like this:
  [a].[costs] < 100 OR [a].Caption LIKE “*service*”
  Meaning: In order to be mapped to the pattern vertex named „a“, the attribute „cost“ of a vertex candidate must carry a value lower than 100, or its name should contain the word „service“.

- With rules, we can relate the properties of any pattern element to each other (e.g., “the name of the edge x must be different from the value of the attribute y of the vertex z”)

CONCRETE SYNTAX

GLOBAL RULES

- Global rules can be composed using the following concepts:

Syntax	Supported elements	Description
[a].Value	Instance Attributes	The value of an Instance Attribute
[a].[b].Value	Objects	The value of an element's Type Attribute
[a].Type	Objects, Instance Attributes, Edges	The type of the element as String
[a].[Caption].Value	Objects, Instance Attributes, Edges	The caption of the element

CONCRETE SYNTAX

GLOBAL RULES

- Concepts can be compared using the following operators

Data Type	Operator	Meaning
String	==	Two equal strings.
	!=	Two non-eqal strings.
	LIKE	Like equal, but allows wildcards. [The wildcard '*' allows any number of arbitrary characters, '?' allows one arbitrary character.]
Number	==	Two equal numbers.
	!=	Two non-equal numbers.
	<	A number less than another.
	<=	A number less than or equal to another.
	>	A number greater than another.
	>=	A number greater than or equal to another.
Boolean	==	Two equal boolean values.
	!=	Two non-equal boolean values.

CONCRETE SYNTAX

GLOBAL RULES

- Global rules also allow aggregate functions on the following expressions

Expression	Description
PREDECESSORS	Neighbors connected with a directed edge that points to the element.
SUCCESSORS	Neighbors connected with a directed edge that points to the neighbor.
UNDIRECTED_NEIGHBORS	Neighbors connected with an undirected edge.
NEIGHBORS	All neighbor nodes.
OUTGOING_EDGES	Directed edges pointing away from the element.
INCOMING_EDGES	Directed edges pointing to the element.
UNDIRECTED_EDGES	Edges that are connected to the element and have no direction.
EDGES	All edges connected to the element.
NODES	Requires that the element is a path, all nodes on that path.

CONCRETE SYNTAX

GLOBAL RULES

- This is how the aggregate functions are used

Function	Example Rule	Description
Count	[a].UNDIRECTED_EDGES.Count()	The number of elements specified. In the example, the number of undirected edges connected to a.
Max	[a].SUCCESSORS.Max([duration])	The maximum value of the attribute among the given elements. In the example, the value of the highest duration among the succeeding neighbors of a.
Min	[a].NEIGHBORS.Min([service_time])	The minimum vlaue of the attribute among the given elements. In the example, the lowest value of service_time among all neighbors of a.
Avg	[p].NODES.Avg([duration])	The average of all attribute values of the specified elements. In the example, the average of all occur-rences of the attribute duration on nodes on path p.
Sum	[p].NODES.Sum([duration])	The sum of all attribute values of the specified elements. In the example, the sum of all occurrences of the attribute duration on nodes on path p.

SEMANTICS (OUTLINE)

- Patterns are mapped against a set of models

- Every pattern vertex is mapped to exactly one vertex of a pattern occurrence
- Every simple edge of the pattern is mapped to exactly one edge of a pattern occurrence
- Every path edge of the pattern is mapped to exactly one path of a pattern occurrence

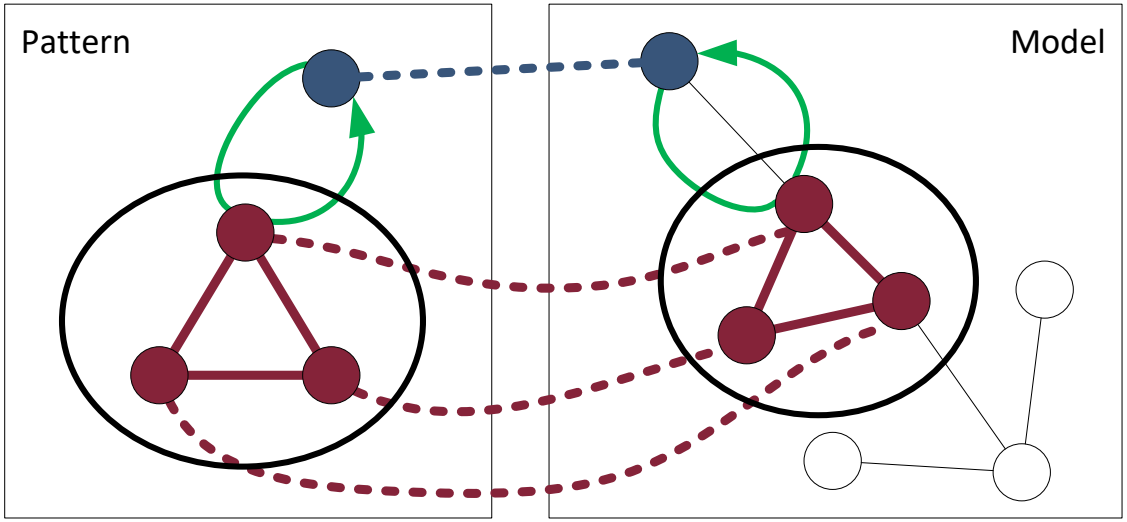
- Hence, the matching algorithm is based on:
  - Subgraph isomorphism considering vertex and edge properties (simple pattern edges)
  - Non-disjunctive, configurable subgraph hoemorphism (path pattern edges) considering vertex and edge properties

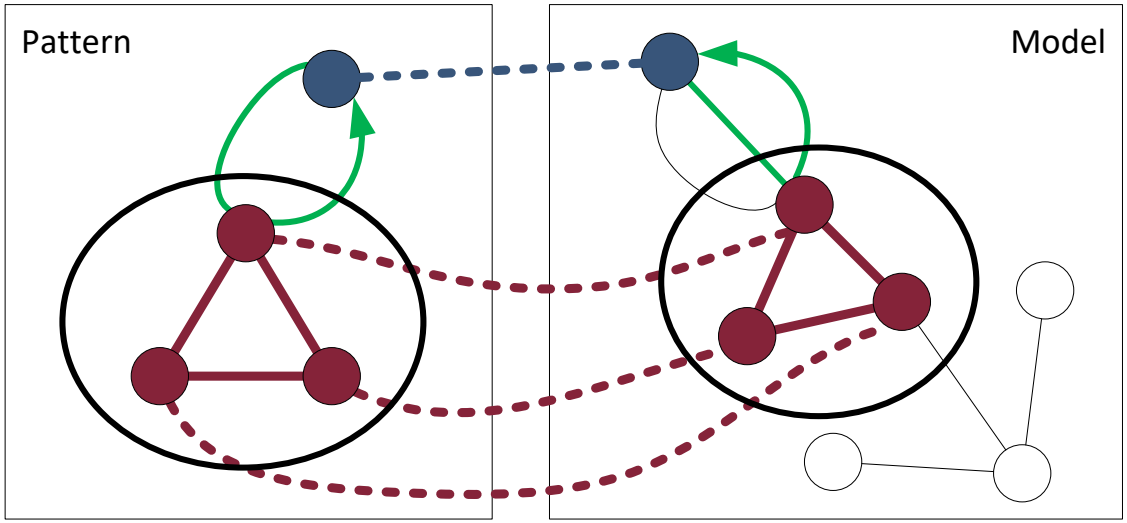
SEMANTICS (IN DETAIL)

→ READING!

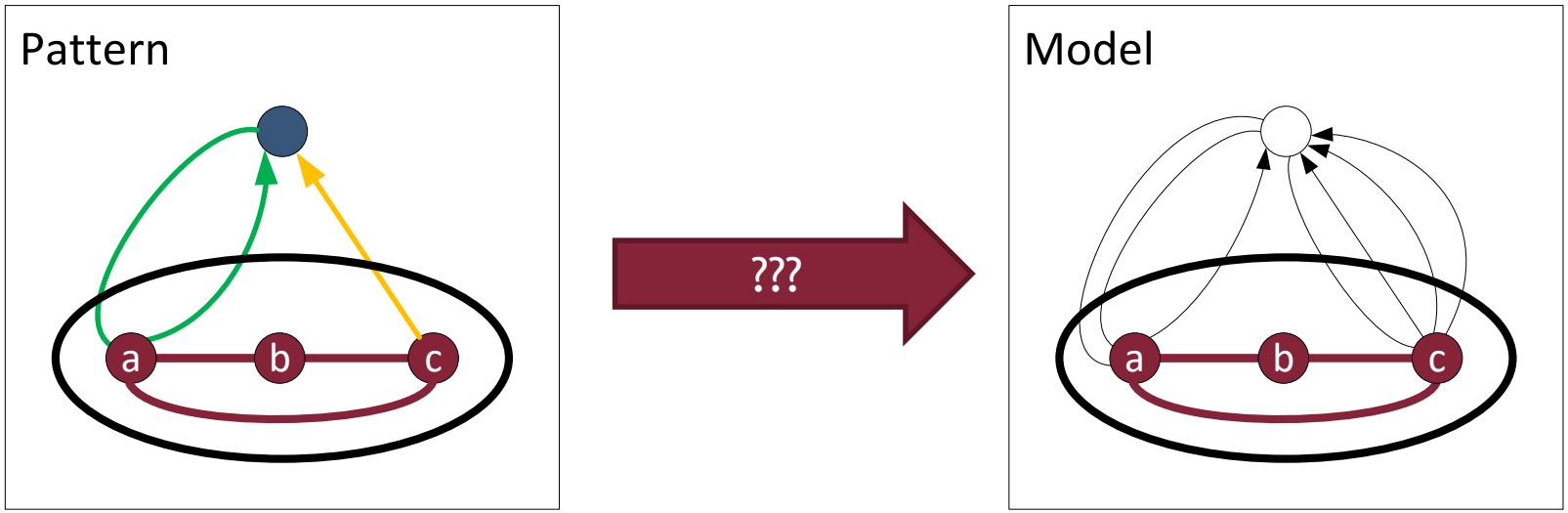
1. $v_{Qx}, v_{Qy} \in V_Q \wedge (v_{Qx}, v_{Qy}, n_Q) \in E_Q \wedge \varepsilon((v_{Qx}, v_{Qy}, n_Q)).minl = \varepsilon((v_{Qx}, v_{Qy}, n_Q)).maxl = 1 \wedge dir = \varepsilon((v_{Qx}, v_{Qy}, n_Q)).dir \Leftrightarrow$
2. $\exists^{=1} v_{ix}' \in V_i', \exists^{=1} v_{iy}' \in V_i' : \alpha_i'(v_{ix}') \in \delta(v_{Qx}).vtypes, \alpha_i'(v_{iy}') \in \delta(v_{Qy}).vtypes, \delta(v_{Qx}).vcaption \sim \chi_i'(v_{ix}'), \delta(v_{Qy}).vcaption \sim \chi_i'(v_{ix}') \wedge$
3. $(\exists^{=1} (v_{ix}', v_{iy}', n_M) \in E_i' : \varepsilon((v_{Qx}, v_{Qy}, n_Q)).ecaption \sim \chi_i'((v_{ix}', v_{iy}', n_M)), org \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).dir, \beta_i'((v_{ix}', v_{iy}', n_M)) \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).etypesr) \vee$
4. $(\exists^{=1} (v_{iy}', v_{ix}', n_M) \in E_i' : \varepsilon((v_{Qx}, v_{Qy}, n_Q)).ecaption \sim \chi_i'((v_{iy}', v_{ix}', n_M)), opp \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).dir, \beta_i'((v_{iy}', v_{ix}', n_M)) \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).etypesr) \vee$
5. $(\exists^{=1} \{v_{ix}', v_{iy}', n_M\} \in E_i' : \varepsilon((v_{Qx}, v_{Qy}, n_Q)).ecaption \sim \chi_i'(\{v_{ix}', v_{iy}', n_M\}), none \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).dir, \beta_i'(\{v_{ix}', v_{iy}', n_M\}) \in \varepsilon(\{v_{Qx}, v_{Qy}, n_Q\}).etypesr),$
6. $v_{Qx}, v_{Qy} \in V_Q \wedge (v_{Qx}, v_{Qy}, n_Q) \in E_Q \wedge \varepsilon((v_{Qx}, v_{Qy}, n_Q)).minl > 1, \varepsilon((v_{Qx}, v_{Qy}, n_Q)).maxl \neq 1 \wedge dir = \varepsilon((v_{Qx}, v_{Qy}, n_Q)).dir \Leftrightarrow$
7. $\exists^{=1} v_{ix}' \in V_i', \exists^{=1} v_{iy}' \in V_i' : \alpha_i'(v_{ix}') \in \delta(v_{Qx}).vtypes, \alpha_i'(v_{iy}') \in \delta(v_{Qy}).vtypes, \delta(v_{Qx}).vcaption \sim \chi_i'(v_{ix}'), \delta(v_{Qy}).vcaption \sim \chi_i'(v_{ix}') \wedge$
8. $\exists^{=1} vertices(path_k(v_{xi}', v_{yi}')_{dir}) \in V_i', \exists^{=1} edges(path_k(v_{xi}', v_{yi}')_{dir}) \in E_i' : \varepsilon((v_{Qx}, v_{Qy}, n_Q)).minl \leq length(path_k(v_{xi}', v_{yi}')_{dir}) \leq \varepsilon((v_{Qx}, v_{Qy}, n_Q)).maxl \wedge$
9. $\varepsilon((v_{Qx}, v_{Qy}, n_Q)).minvo \leq vovl(path_k(v_{xi}', v_{yi}')_{dir}) \leq \varepsilon((v_{Qx}, v_{Qy}, n_Q)).maxvo \wedge \varepsilon((v_{Qx}, v_{Qy}, n_Q)).mineo \leq eovl(path_k(v_{xi}', v_{yi}')_{dir}) \leq \varepsilon((v_{Qx}, v_{Qy}, n_Q)).maxeo \wedge$
10. $\forall t \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).vtypesr \exists v \in vertices(path_k(v_{xi}', v_{yi}')_{dir}) \mid \alpha_i'(v) = t \wedge \exists v \in vertices(path_k(v_{xi}', v_{yi}')_{dir}) \mid \alpha_i'(v) \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).vtypesf \wedge$
11. $\forall t \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).etypesr \exists e \in edges(path_k(v_{xi}', v_{yi}')_{dir}) \mid \beta_i'(e) = t \wedge \exists e \in edges(path_k(v_{xi}', v_{yi}')_{dir}) \mid \beta_i'(v) \in \varepsilon((v_{Qx}, v_{Qy}, n_Q)).etypesf \wedge$
12. $((vertices(path_k(v_{xi}', v_{yi}')_{dir}) \cup edges(path_k(v_{xi}', v_{yi}')_{dir}) \cap M_n'(\theta.Q) = M_p'(\theta.Q) \mid \theta.constraint = pre$

MAPPING EXAMPLES

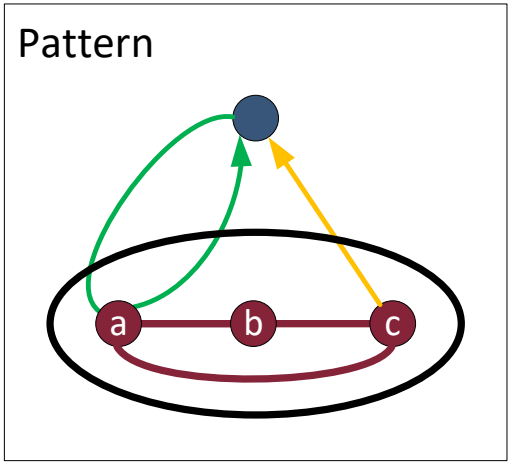




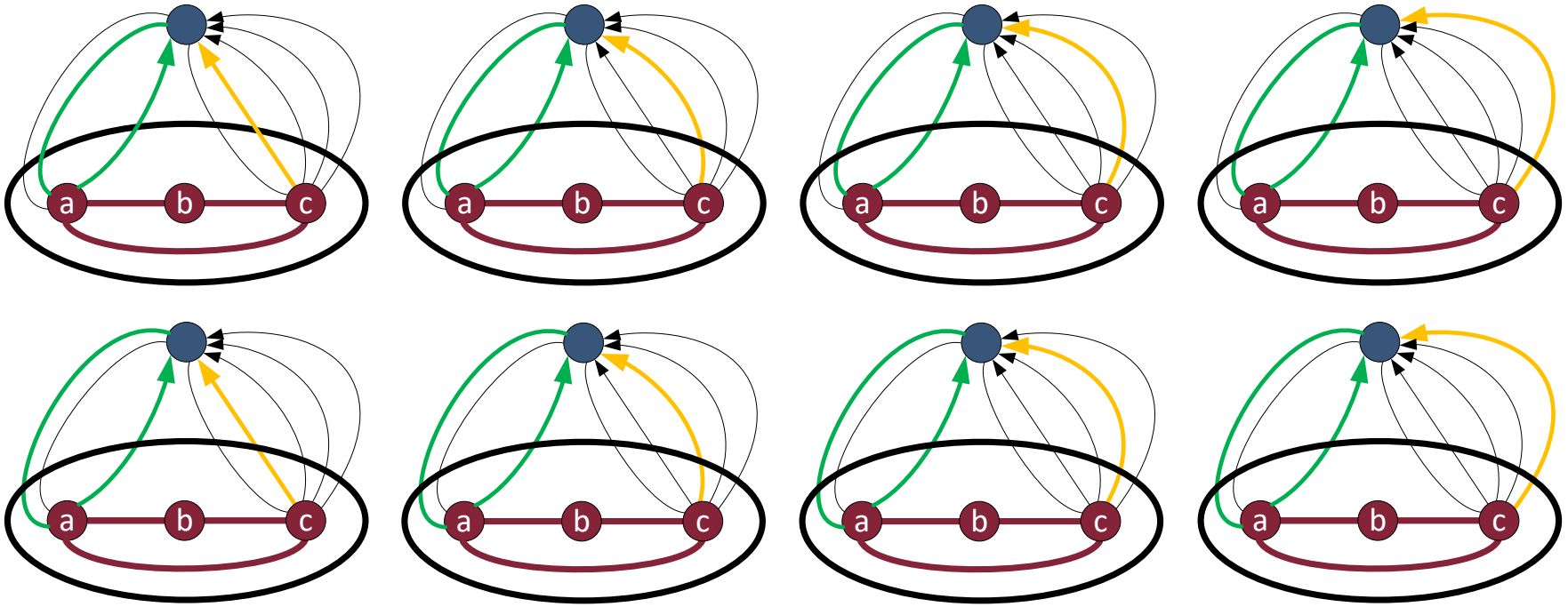
MAPPING EXAMPLES



MAPPING EXAMPLES



Model Mapping Possibilities



SIDE NOTE

- A new version of DMQL also includes concepts to easier handle execution semantics of process models
  - Forbidden vertices
  - Forbidden edges
  - Forbidden paths

- For instance, a vertex x that is connected to a forbidden vertex y means that a mapping candidate for x is mapped only if it is NOT connected to another vertex that matches the properties of y

AGENDA

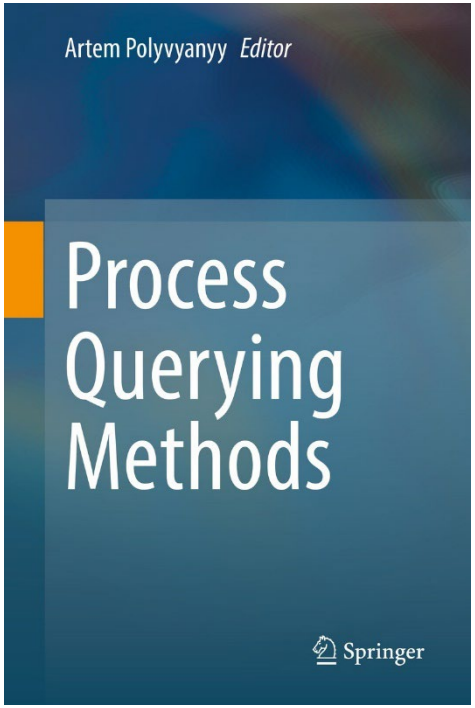
- Motivation for a Diagramed Model Query Language (DMQL)
- Requirements of DMQL
- The Diagramed Model Query Language (DMQL) incl. Parallel Tool Presentation
- Further Information on the Topic Model Query

READING ON DMQL

- Delfmann, P.; Breuker, D.; Matzner, M.; Becker, J.: Supporting Information Systems Analysis Through Conceptual Model Query – The Diagramed Model Query Language (DMQL). *Communications of the Association for Information Systems (CAIS) 37 (2015) 1*, pp. 473-509.

OTHER MODEL QUERY LANGUAGES

- aFSA
- APQL
- BeehiveZ
- BPMN VQL
- BPMN-Q
- BPQL
- BP-QL
- Cypher (Neo4j)
- EMF-IncQuery
- IPM-PQL
- OCL
- PPSL
- SPARQL
- Untanglings
- VMQL



- Book on Process Query Languages
- Also: see Related Work Sections of GMQL and DMQL Readings

UNIVERSITÄT KOBLENZ · LANDAU

IWVI

BUSINESS PROCESS MANAGEMENT

MODEL QUERY III:
THE DIAGRAMED MODEL QUERY LANGUAGE (GRAPH MATCHING)

INSTITUTE FOR IS RESEARCH

www.uni-koblenz.de