Diagramatic Software Specification

Adrian Rutle, Yngve Lamo (HiB)
Uwe Wolter (UiB)
Norway

Introduction

- Need for a common language for
  - Designers
  - Programmers
  - Experts of the universe
  - …
- In order to:
  - reason about the problem
  - verify business logic
- Graphical language is well suited for this task

Why a graphical language?

- Simple
  - Easier to understand compared to textual languages
- Universal
  - Different background → same understanding

What's a good modelling language?

- Graph-based,
- has a formal interpretation and
- suffciently expressive (to capture all the peculiarities of the universe)

Proposal: GENERALIZED SKETCHES

State-of-the-art

For specifications in software engineering:

- Many graphical modeling languages
- Few of them have proper semantics

→ Specifications that are difficult to maintain due to:

- ambiguous constructions
- semantic relativism

Our contribution to the field

- Use of Generalized Sketches (GS) to:
  - compare
  - unify
  - integrate

  diagrammatic specification languages
- Achieve this by:
  - Mapping different modeling languages to the GS format
  - Superimposing signatures from different graphical notations
- Graphical notations are parameterized by signatures
- *What* should be modeled must be standardized
- BNF is an analogy:

  a formal notation to describe the syntax of a given language

Generalized Sketches: history

- Ehresmann's sketches: graphical presentations of categories (1968)
- Makkai: (1995) diagram predicates + sketches → Generalized Sketches.
- Diskin (1997) made GS:
  - more direct
  - more applicable to SE
  - complex diagram operations (operational ske.)

Formal definitions

- Graph: $G = (G_0, G_1, src^G, trg^G)$
- Graph homomorphism

$$h : G \to H$$
$$h_0 : G_0 \to H_0$$
$$h_1 : G_1 \to H_1$$
$$\forall e \in G_1 :$$
$$src^H(h_1(e)) = h_0(src^G(e)) \; and$$
$$trg^H(h_1(e)) = h_0(trg^G(e))$$

- Diagram: $\delta : Shape \to G$
- Signature: $\Sigma = < P, Arity(P) >$
- Sketch: $(\Sigma -)sketch\ S = < G_S, D_S(P) >$

$$where\ \{\delta : Arity(P) \to G_S\} \in D_S(P)$$

Sketch morphism

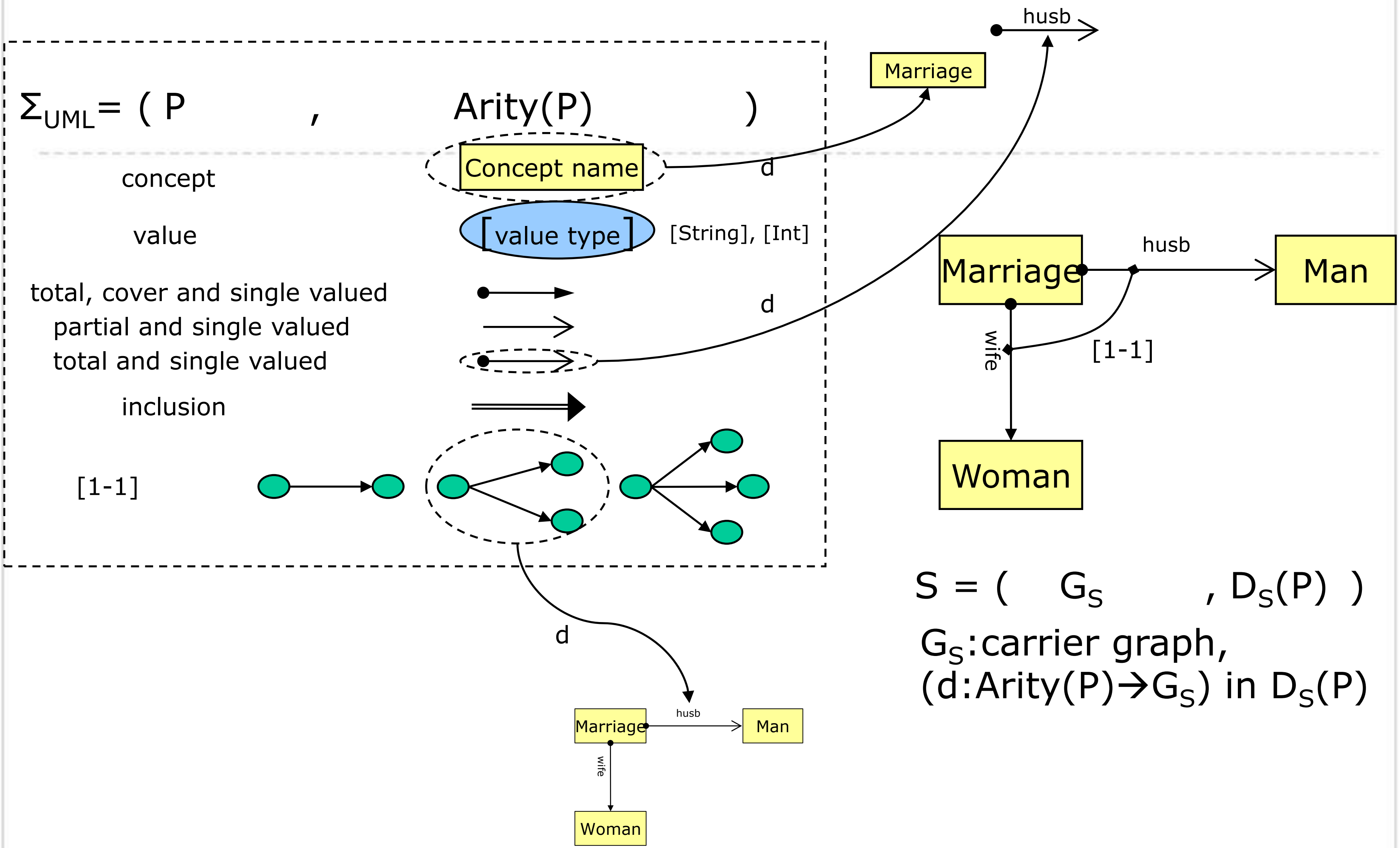
$$A\ sketch\ morphism\ \mu : S_1 \to S_2$$

$$is\ a\ graph\ homomorphism\ \mu : G_{S_1} \to G_{S_2}$$

$$such\ that\ \forall \delta : Arity(P) \to G_{S1} \in D_{S_1}$$

$$\Rightarrow \delta ; \mu : Arity(P) \to G_{S_2} \in D_{S2}$$

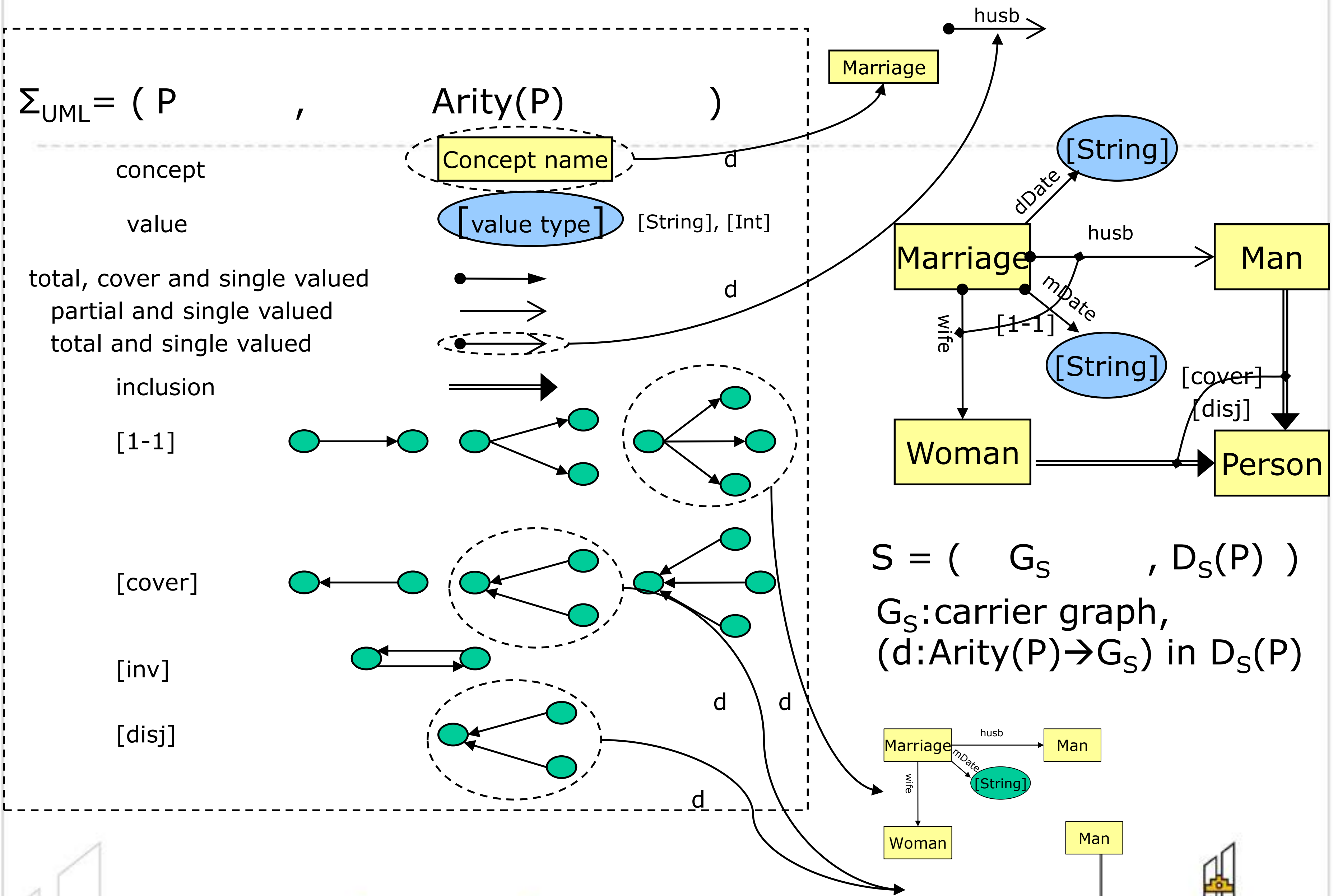
$\Sigma_{UML} = ($ P , Arity(P) $)$



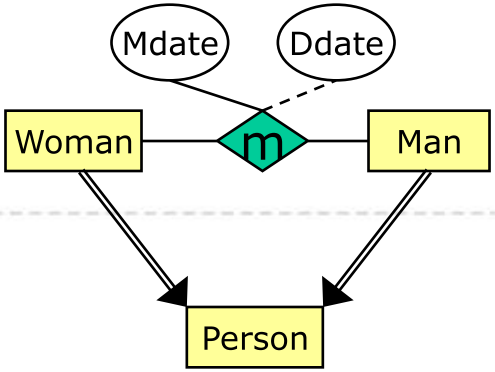
$S = (\; G_S \; , D_S(P) \;)$

G_S:carrier graph,

$(d{:}Arity(P) \rightarrow G_S)$ in $D_S(P)$


$\Sigma_{UML} = ($ P , Arity(P) $)$
concept
Concept name
d
Marriage
value
[value type]
[String], [Int]
total, cover and single valued
partial and single valued
total and single valued
d
husb
inclusion
[1-1]
[cover]
[inv]
[disj]
d
d
d
[String]
dDate
Marriage
husb
Man
mDate
wife
[1-1]
[String]
[cover]
[disj]
Woman
Person
$S = (G_S , D_S(P))$
G_S:carrier graph,
$(d: Arity(P) \rightarrow G_S)$ in $D_S(P)$
Marriage
husb
Man
mDate
wife
[String]
Woman
Man
Woman
Person

$\Sigma_{ER} = ($ P , Arity(P) $)$



concept — Consept name — r

value — value — Mdate, Ddate

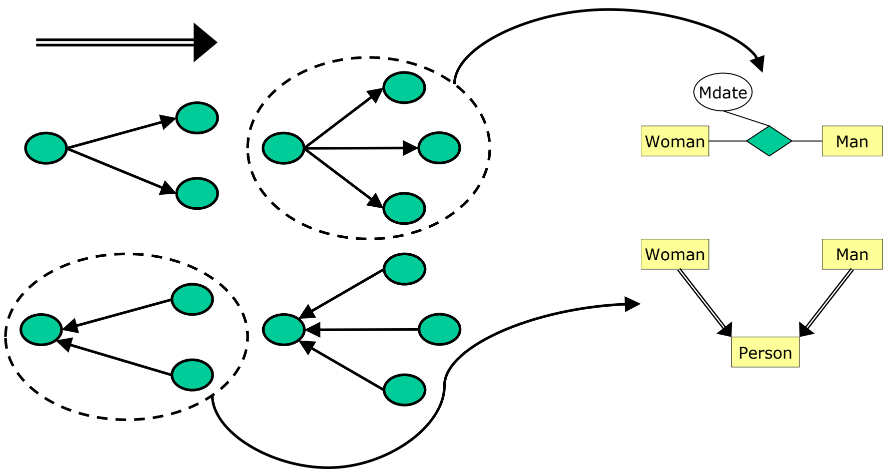
Partial mapping - - - - - - - - -

Total mapping ———

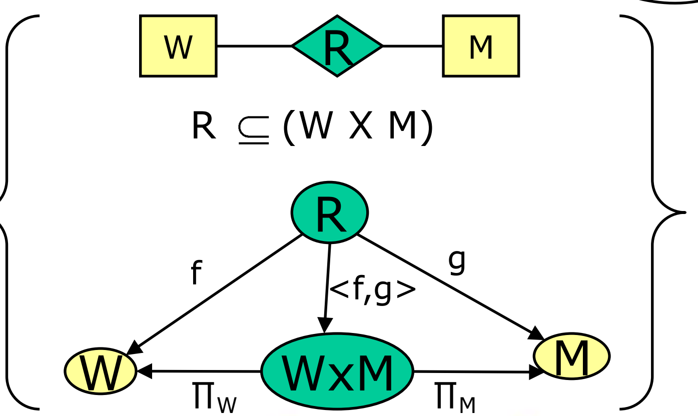
inclusion ⟹

Jointly mono:
Binary relation
Ternary relation …

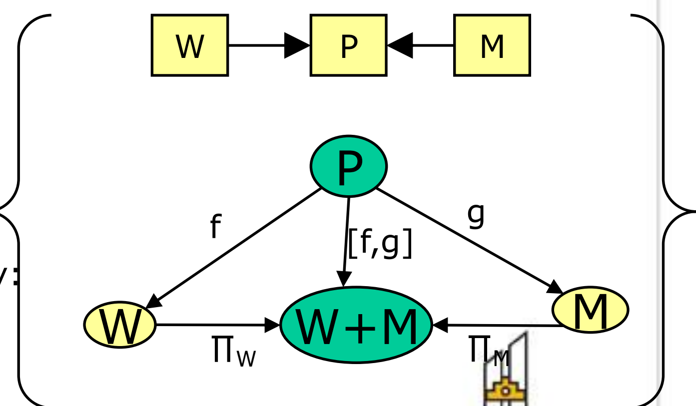
Disjoint union:
(inclusion)

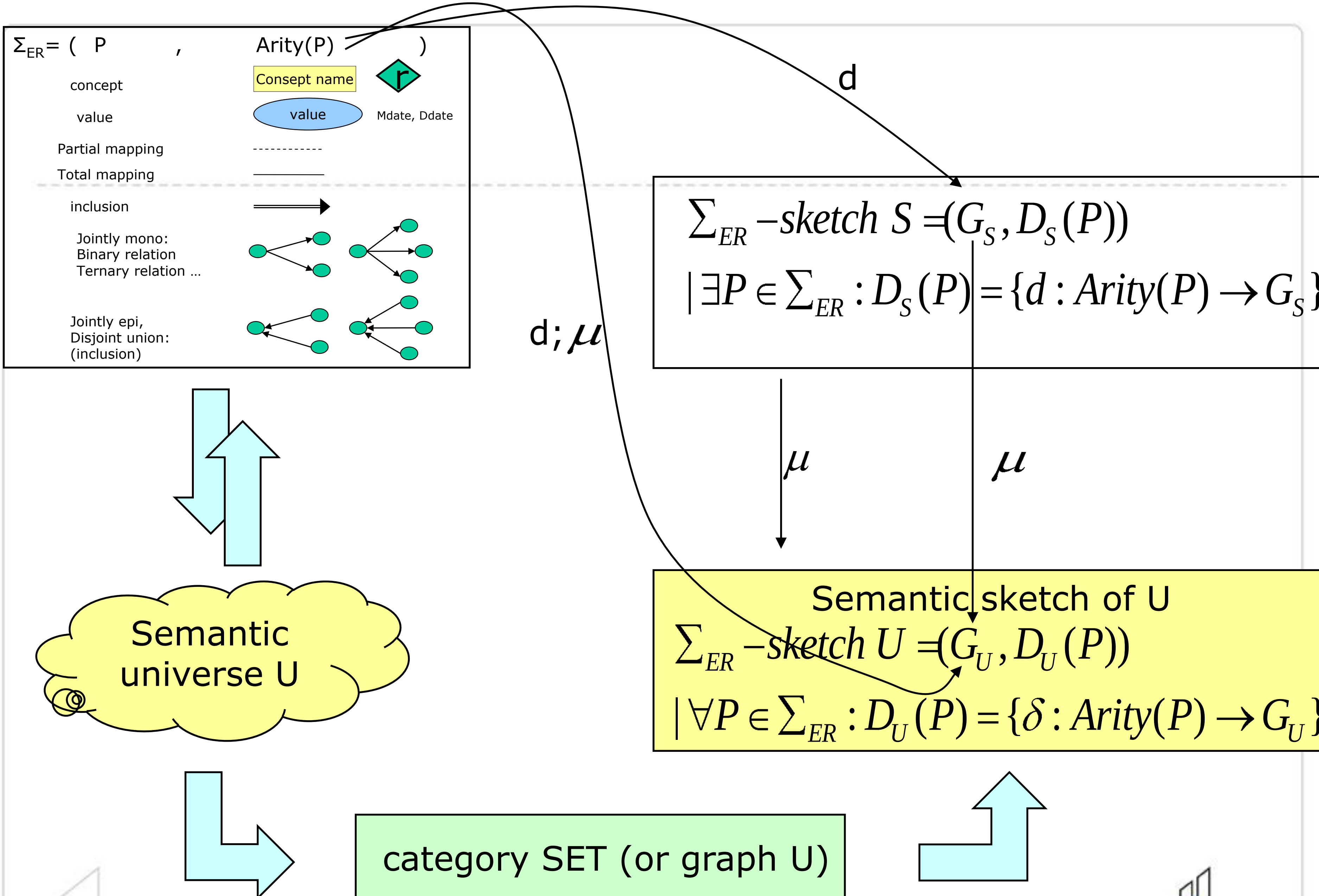


Equivalent ways for expressing binary relations. Categorically: product

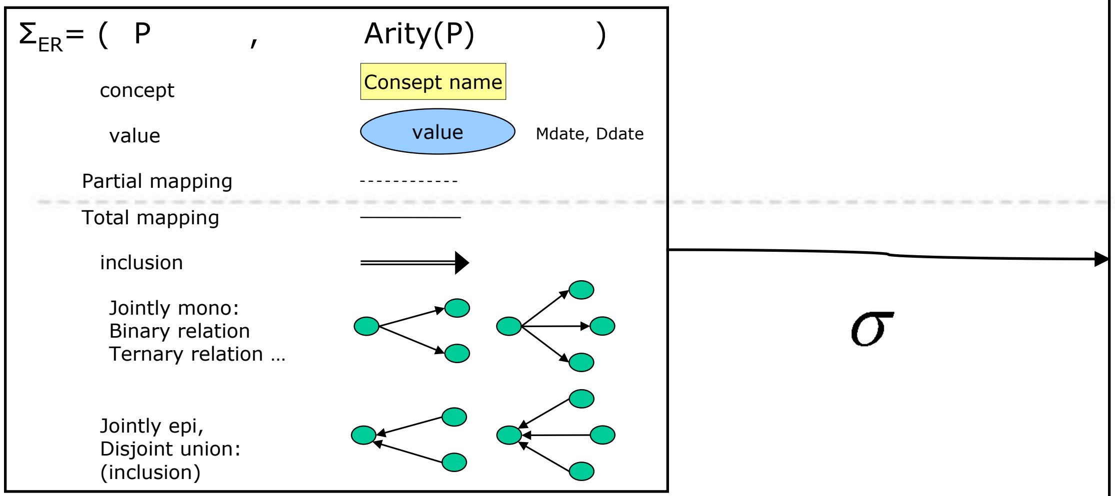


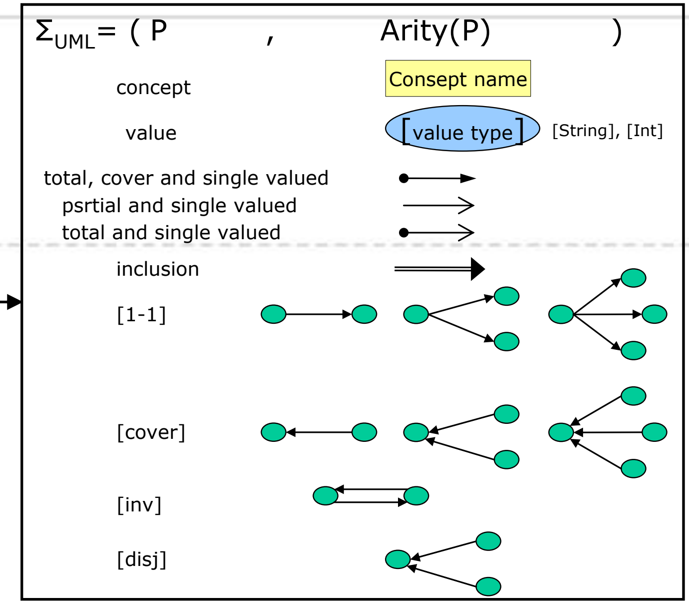
Equivalent ways for expressing inclusion. Categorically: co-product





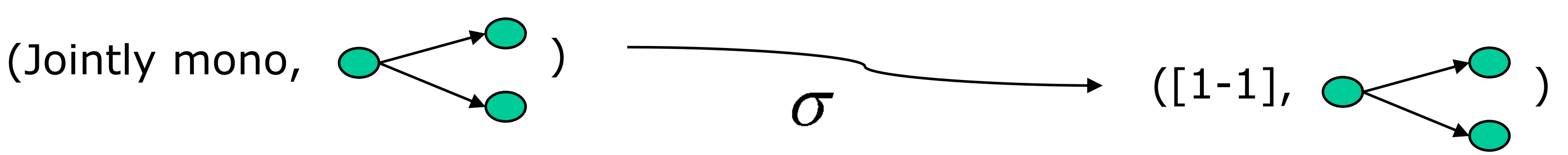
Σ_{ER} = (P , Arity(P))
concept
Consept name
r
value
value
Mdate, Ddate
Partial mapping
Total mapping
inclusion
Jointly mono:
Binary relation
Ternary relation …
Jointly epi,
Disjoint union:
(inclusion)
d
d; μ
Σ_{ER} −sketch $S = (G_S, D_S(P))$
$\mid \exists P \in \Sigma_{ER} : D_S(P) = \{d : Arity(P) \to G_S\}$
μ
μ
Semantic universe U
Semantic sketch of U
Σ_{ER} −sketch $U = (G_U, D_U(P))$
$\mid \forall P \in \Sigma_{ER} : D_U(P) = \{\delta : Arity(P) \to G_U\}$
category SET (or graph U)

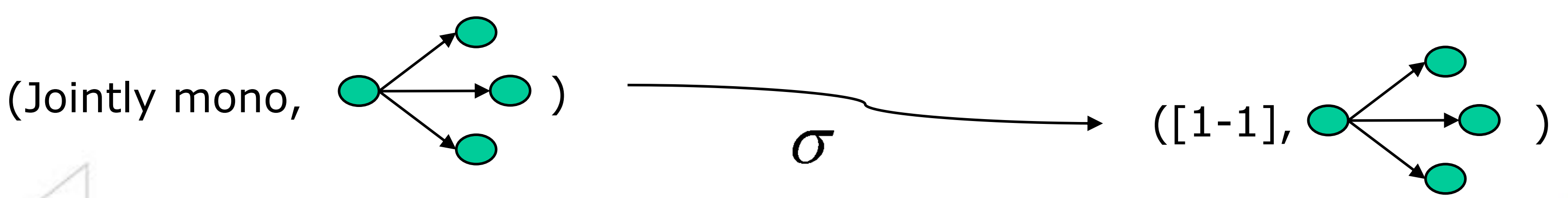




Such that:

$$\forall P \in \sum_{ER} : Arity(P) \subseteq Arity(\sigma(P))$$





Our objectives (theory)

- Formalization of:
  - ER-diagrams,
  - UML-diagrams,
  - DB-schemes and
  - Ontology languages,

  by GS
- Investigation of the
  - integration,
  - combination and
  - modularization

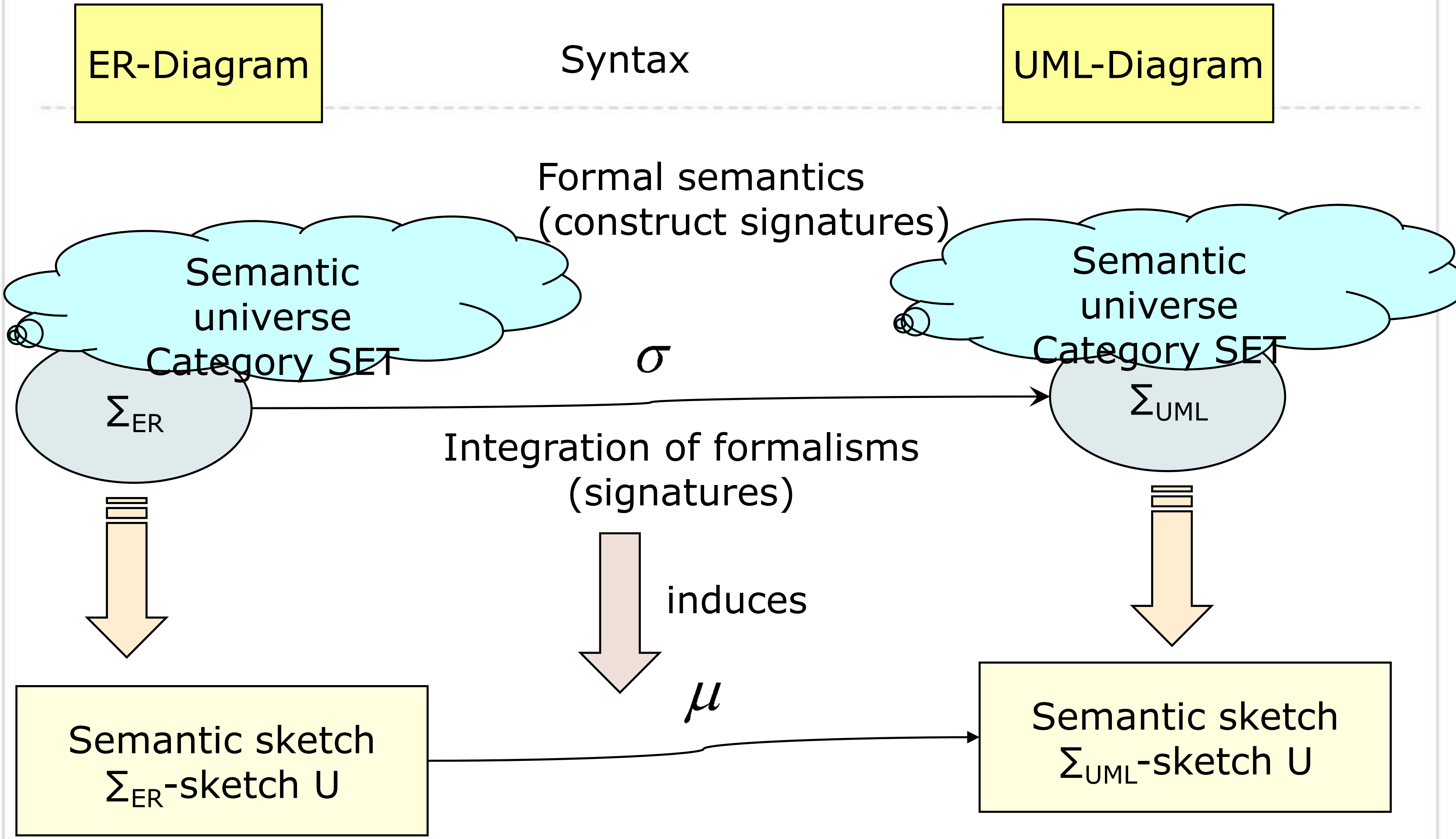
  of specifications within single specfication formalism based on GS

Our objectives (technical)

- Design and development of tools supporting the application of GS in the field of software engineering:
  - drawing UML and ER diagrams based on GS formalism
  - support for mappings/translations between the two types of diagrams by mapping their signatures
  - code-generator
    - programming languages
    - specification language

    based on their graphical specications
  - case studies to evaluate the theory in practice.

Summary


ER-Diagram
Syntax
UML-Diagram
Formal semantics
(construct signatures)
Semantic
universe
Category SET
Semantic
universe
Category SET
σ
Σ_{ER}
Σ_{UML}
Integration of formalisms
(signatures)
induces
μ
Semantic sketch
Σ_{ER}-sketch U
Semantic sketch
Σ_{UML}-sketch U

Thank you!

What's done and progress:

- Diskin's program
- Ørjan Hatlands Contribution
- New Master-thesis by Stian Skjerveggen

Facts

Sketch-able spec. ←→ Formal spec.

Diskin and Kadish:

- Graphical specifications (ER diagram or UML class diagram,) can be considered as:
  - abbreviations or
  - visualizations

of sketches for a fixed signature

where the signature is the corresponding diagram type

Diskin and Kadish:

- A graphical notation: a visualization on the top of a specification core
  - Visualization: a presentation (a user interface) of specication,
  - Specification ”deals with” semantics of the notational constructs