Generalized Sketches and Indexed vs. Fibred Semantics

Uwe Wolter

University of Bergen, Norway

IMAR, November 18th 2009, Bucharest, Romania
Outline

1. Introduction and Motivation
2. Sketches and Generalized Sketches
3. Generalized Sketches in view of Institutions
4. Diagram Predicate Framework
State of the Art

- Diagrammatic, i.e., graph-based, modeling techniques like UML-diagrams, ER-diagrams, ORM-diagrams, database schemata, ontologies, XML, ..., are widely used in software engineering (SE).
- Until recently diagrammatic models were mainly used as a communication medium between business experts, software designers and programmers, and their precision and semantics was (although very desirable) not a must.
Dramatic changes

Few years ago started a rapid invasion of the model-centric trends in software industry (MDE, MDD, MDA™). These activities aim at making models rather than code the primary artifacts of software development with code to be generated directly from models.

New needs

For MDE and friends, having a precise formal semantics for the diagrammatic notations they employ is an undisputable must. An overwhelming amount of formal semantics for diagrammatic languages in use has been proposed. Most of them employ the familiar first-order or similar logical systems, and fail to do the job because of unfortunate mismatch between the logical machineries they use for formalization (string-based) and the internal logics of the domains they intend to formalize (graph-based).
**Question**

Does there exist something in mathematics that could be called a "graph-based specification formalism"?

**Answer**

Yes, the sketch formalism developed within Category Theory (CT).
- Introduced 1968 by Ehresmann.
- Re-formulated and re-launched in the 80s by Barr and Wells for application in computer science (CS).
- Extremely expressive.
**Definition of sketch**

A **sketch** $\mathcal{S} = (G^\mathcal{S}, \Delta^\mathcal{S})$ consists of a graph $G^\mathcal{S}$ and a set $\Delta^\mathcal{S}$ of **diagrams** $(d, I)$ in $G^\mathcal{S}$, i.e., of graph homomorphisms $d : I \rightarrow G^\mathcal{S}$. Moreover, each diagram in $\Delta^\mathcal{S}$ is labeled either as a commutative diagram or a limit cone or a colimit cocone.

**Remark**

Sketches can be classified according to the structure of the shapes $I$ and the labels used. Examples are finite product sketches and finite limit sketches (left exact sketches).
Algebraic signature for natural numbers

We declare one sort \( N \) and three operations \( z : \rightarrow N, s : N \rightarrow N, a : N N \rightarrow N \).

The corresponding finite product sketch

Underlying graph:

Product cones:
Introduction and Motivation
Sketches and Generalized Sketches
Generalized Sketches in view of Institutions
Diagram Predicate Framework

Sketches - Example Diagrammatic Modeling

A sample of sketching a diagrammatic notation

(a1) ER-diagram

House

property

Ownership

owner

date

dd mm yy

Person

(a2) UML class diagram

House

owns 0..*

isOwned 1

Person

(a) Engineering data models

(b) Classical sketch specification (b1 and b2)

(c) Generalized sketches (c1 and c2)

Uwe Wolter

Generalized Sketches and Indexed vs. Fibred Semantics
Why Generalized Sketches?

Inherent drawbacks of sketches in practical applications

- Auxiliary elements have to be introduced.
- Some original elements can be only derived.
- Simple and important properties like mono, epi have to be coded.

⇒ For software engineers this looks artificial, unnecessary, and misleading.

- There are practical relevant predicates that cannot be expressed by limits and colimits, at all. Bear in mind that the categories \textbf{Rel} and \textbf{Mult}, respectively, have no limits.
Why Generalized Sketches?

Some History

Generalized sketches have been developed in the 90’s in parallel by Makkai and a group in Latvia (Diskin, Cadish, et.al.)

- Makkai developed an abstract categorical approach to completeness theorems in logic.
- The Latvian group developed generalized sketches in the context of data modeling and applied them in a few industrial projects.

General in two aspects

1. Arbitrary ”base category” **Base** instead of category **Graph**.
2. Arbitrary ”user defined” predicates instead of ”commutative”, ”limit”, ”colimit” only.

**Attention**: The semantics of the predicates ”commutative”, ”limit”, ”colimit” is pre-defined in any category. In contrast, the semantics of a ”user defined” predicate has to be defined explicitly by its inventor.
**Generalized Sketches - Example Diagrammatic Modeling**

A sample of sketching a diagrammatic notation

- (a1) ER-diagram

- (a2) UML class diagram

- (a) Engineering data models

- (b) Classical sketch specification (b1 and b2)

- (c) Generalized sketches (c1 and c2)

Uwe Wolter  Generalized Sketches and Indexed vs. Fibred Semantics
Definition of Notation

A (diagram predicate) notation $\mathcal{N} = (\Pi, \alpha)$ over a category $\text{Base}$ is given by a set $\Pi$ of predicate symbols and a map $\alpha : \Pi \to \text{Base}_0$. For a predicate symbol $P \in \Pi$, the $\text{Base}$-object $\alpha(P)$ is called the arity of $P$.

Definition of Specification

An $\mathcal{N}$-specification ($\mathcal{N}$-sketch) $S = (G^S, \Delta^S)$ consists of a $\text{Base}$-object $G^S$ and a set $\Delta^S$ of constraints, i.e., of labeled diagrams $(P, d)$ in $G^S$ with $d : \alpha(P) \to G^S$ a morphism in $\text{Base}$. 
Specification Morphisms

An **morphism** $f : S \rightarrow S'$ between $\mathcal{N}$-specifications $S = (G^S, \Delta^S)$ and $S' = (G^{S'}, \Delta^{S'})$ is a morphism $f : G^S \rightarrow G^{S'}$ in **Base** preserving constraints, i.e., for all constraints $(P, d) \in \Delta^S$ we have that $(P, d; f) \in \Delta^{S'}$.

\[ \alpha(P) \xrightarrow{d} G^S \xrightarrow{d; f} G^{S'} \xrightarrow{f} \]

$\mathcal{N}$-specifications and morphisms constitute a category $\text{Ske}(\mathcal{N})$.

**Remark:** Makkai works with a hierarchy of sketches starting with the base category $\text{Graph}$. 
Question
Can we present generalized sketches as institutions?

Answer
Yes! Given a notation $\mathcal{N}$ we get an institution for any semantic universe that allows us to define a corresponding "indexed semantics". In case $\textbf{Base}$ has pullbacks, we can define a "fibred semantics" (within $\textbf{Base}$) and get a "pseudo institution" instead.
Introduction and Motivation
Sketches and Generalized Sketches
Generalized Sketches in view of Institutions
Diagram Predicate Framework

Signatures and Formulas?

Category of signatures

Given an \( \mathcal{N}\)-specification (\( \mathcal{N}\)-sketch) \( S = (G^S, \Delta^S) \) it is quite natural to consider the underlying Base-object \( G^S \) as a signature and the constraints in \( \Delta^S \) as formulas/sentences. So, our category of signatures is the category Base! (Remember the idea of ”signature graphs” in Algebraic Specifications!)

Formulas

Given a ”signature”, i.e., a Base-object \( G \), we denote by

\[
Fm(G) = \{(P, d) \mid P \in \Pi, d \in \text{Base}(\alpha(P), G)\}
\]

the set of all formulas over \( G \).
Any morphism \( s : G \to G' \) in Base defines a functor between slice categories

\[
s_* : (\text{Base} \downarrow G) \to (\text{Base} \downarrow G')
\]

by simple post-composition, i.e., by \( s_* (A, d) = (A, d; s) \) for all objects \((A, d : A \to G)\) in \((\text{Base} \downarrow G)\) and by \( s_* (f) = f \) for all morphisms \( F : (A, d) \to (B, e) \) in \((\text{Base} \downarrow G)\).
Translation maps and Sentence Functor

Any morphism $s : G \rightarrow G'$ in $\text{Base}$ defines a translation map $Fm(s) : Fm(G) \rightarrow Fm(G')$ with $Fm(s)(P, d) = (P, s_*(d)) = (P, d; s)$. Since composition in $\text{Base}$ is associative

\[
\begin{array}{c}
\alpha(P) \\
\downarrow d \\
\downarrow d; s \\
G \quad s \quad G' \quad s' \quad G'' \\
\downarrow \\
\end{array}
\]

the assignments $G \mapsto Fm(G)$ and $s \mapsto Fm(s)$ define a **sentence functor** $Fm : \text{Base} \rightarrow \text{Set}$.

**Remark**: Actually we consider also ”dependencies” between predicates, i.e., $\Pi$ is a category and $\alpha : \Pi \rightarrow \text{Base}$ a functor. In this case, we get a sentences functor $Fm : \text{Base} \rightarrow \text{Cat}$.
What do I mean by "indexed"?

- For mathematicians/theoreticians it is quite natural to describe the semantics of a formalisms in an "indexed manner", i.e., as an interpretation of syntactic entities in a "semantic universe" $\mathcal{U}$.

- "Indexed semantics" is omnipresent in Universal Algebra, Algebraic Specifications, Denotational Semantics, Categorical Logic/Algebra, ..., where $\textbf{Set}$ is often the chosen semantic universe.

- "Indexed semantics" supports a clear separation of syntax and semantics, and until 2-3 years ago this was the only way I could think about semantics.
Indexed Semantics - Informally

Indexed semantics for sketches

1. To define an indexed semantics for sketches we have to choose the base category \( \text{Base} \) and the semantic universe \( \text{U} \) in such a way that the objects in \( \text{Base} \) can be interpreted in \( \text{U} \).

2. If we choose, e.g., \( \text{Base} = \text{Graph} \) and \( \text{U} = \text{Set} \) we can define an interpretation (a "model") of a signature (graph) \( G \) as a graph homomorphism \( m : G \rightarrow \text{Set} \), i.e., the category of "models" can be given by the "functor category" \( \text{mod}(G) = [G \rightarrow \text{Set}] \).

3. Any graph homomorphism \( s : G \rightarrow G' \) defines then a functor \( \text{mod}(s) : \text{mod}(G') \rightarrow \text{mod}(G') \) by simple pre-composition, i.e., by \( \text{mod}(s)(m') = s; m' \) for all "models" \( m' : G' \rightarrow \text{Set} \).
Indexed Semantics - Informally

Indexed semantics - model functor

- Again, due to associativity of composition in \( \text{GRAPH} \), the assignments \( G \mapsto \text{mod}(G) \) and \( s \mapsto \text{mod}(s) \) define a "model" functor \( \text{mod} : \text{Graph}^{\text{op}} \to \text{Cat} \).

- Also the satisfaction condition can be shown straightforwardly.

Summary/Claim

For any choice of \( \text{Base} \) and \( U \), that allows to interpret \( \text{Base} \)-objects in \( U \) compatible with composition in \( \text{Base} \), it is relatively straightforward to show that the triple \((\text{Base}, N, U)\) provides an institution.
Reality in Software Engineering

BUT !!!
People think differently!

- Software Engineers and especially people in OO-modeling think completely different!
- They think rather in terms of "instances" then of "interpretations (models)"!
- And, even worth, they call a model what we call a specification!

⇒ Note, that this is in accordance with the use of the term "model" in physics, chemistry, and many other areas.
- They iterate the model-instance pattern and call it then metamodel, meta-metamodel, ...
Fibred semantics - Example of instances

(c1) Valid instance of schema $S$

(c2) Invalid instance of schema $S$

(a) Signature, $\Pi$

(b) Schema/ specification/model/sketch, $S$
## Fibred semantics - MOF-based modelling hierarchy

<table>
<thead>
<tr>
<th>Modelling layers</th>
<th>Standards and examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_3$: Meta-metamodel</td>
<td>MOF</td>
</tr>
<tr>
<td>$M_2$: Metamodel</td>
<td>UML language: Class, Attribute</td>
</tr>
<tr>
<td>$M_1$: Model</td>
<td>A UML model: Class Person with Attributes name and address</td>
</tr>
<tr>
<td>$M_0$: Instance</td>
<td>An instance of Person: “Ola Nordmann” living in “Sotraveien 1, Bergen”</td>
</tr>
</tbody>
</table>
Main ideas and features

- Syntactic entities (models) and semantic entities (instances) live in the same universe.
  ⇒ This makes it difficult to keep track and to explain to software engineers the difference between syntax and semantics.

- An instance of a syntactic entity (model) is given by a semantic entity and an "abstraction" morphism from the semantic entity into the model.

- It’s quite natural to iterate the model-instance pattern.
  ⇒ This means that the roles "syntax" and "semantics" have only a relative meaning.

- Such a natural iteration is not possible with indexed semantics!
Introduction and Motivation
Sketches and Generalized Sketches
Generalized Sketches in view of Institutions
Diagram Predicate Framework

Fibred Semantics - Formally

General Assumption
Let be given a (diagram predicate) notation \( \mathcal{N} = (\Pi, \alpha) \) over a category \( \textbf{Base} \) with \( \alpha : \Pi \rightarrow \text{Base}_0 \).
We assume that the base category \( \textbf{Base} \) has pullback.

Semantics of notations
A semantic interpretation of the notation \( \mathcal{N} \) is a mapping \( \llbracket \ldots \rrbracket \), which assigns to each predicate symbol \( P \in \Pi \) a set \( \llbracket P \rrbracket \subset \{ \tau \in \text{Base}_1 \mid \text{cod}(\tau) = \alpha(P) \} \) of valid instances, where \( \llbracket P \rrbracket \) is assumed to be closed under isomorphisms.
Semantic of Signatures

For any signature, i.e., any Base $G$ the category of instances is defined as slice category $\text{Inst}(G) = (\text{Base} \downarrow G)$.

Satisfaction Relation

An instance $\tau : O \rightarrow G$ of $G$ satisfies a formula $(P, d)$ over $G$, written $\tau \models (P, d)$, iff $\tau^* = PB_d(\tau) \in \llbracket P \rrbracket$.

\[
\begin{array}{c}
O^* \xrightarrow{d^*} O \\
\tau^* \xrightarrow{\alpha(P)} [PB] \\
\alpha(P) \xrightarrow{d} G
\end{array}
\]
**Forgetful Functors by pullbacks**

Any morphism $s : G \rightarrow G'$ in $\text{Base}$ induces a functor $s^* : (\text{Base} \downarrow G') \rightarrow (\text{Base} \downarrow G)$ right-adjoint to the functor $s_* : (\text{Base} \downarrow G) \rightarrow (\text{Base} \downarrow G')$, where $s^*(\tau') = PB_s(\tau)$ for any instance $\tau' : O' \rightarrow G'$ of $G'$.

**Satisfaction Condition = composition + decomposition of pullbacks**

For any morphism $s : G \rightarrow G'$, any instance $\tau' : O' \rightarrow G'$ of $G'$ and any formula $(P, d)$ over $G$ we have

$$s^*(\tau') \models_G (P, d) \iff \tau' \models_{G'} (P, d; s).$$
Why not an institution?

Pullbacks are only determined up to isomorphism. Therefore, for morphisms $s : G \to G'$ and $s' : G' \to G''$ the functors $(s; s')^*$ and $s^*; s'^*$ will be not equal but only natural isomorphic. One can, however, prove that the family of those natural isomorphisms satisfies certain coherence conditions.

Pseudo Functor

The assignments $G \mapsto (\text{Base} \downarrow G)$ and $s \mapsto s^*$ define a pseudo functor $\text{Inst} : \text{Base}^{op} \to \text{Cat}$. 
Why the name "fibred"?

For any indexed category, i.e., any functor $F : C^{op} \to \text{Cat}$ the Grothendieck construction provides a category $\text{Flat}(F)$ together with a projection functor $pr_F : \text{Flat}(F) \to C$.

The concept of (split) fibration can be seen as an axiomatization of those functors that can be obtained by the Grothendieck construction.
**Summary/claim**

Any formalism with fibred semantics will not provide a ”model functor” but only a pseudo ”model functor”.

**Questions**

- Is the present, well-developed, theory of institutions not applicable to fibred semantics?
- Should we start to extend the theory to pseudo model functors and maybe even pseudo sentence functors?
- Or should we better reconstruct the whole theory in terms of fibrations?
Diagram Predicate Framework (DPF)

DPF - Idea

By Diagram Predicate Framework we name a research activity and an informal research group aimed at to apply generalized sketches in software engineering and to develop further the theory according the practical needs.

DPF - Persons

- Uwe Wolter, Associate Professor, University of Bergen
- Alessandro Rossini, PhD-student, University of Bergen
- Yngve Lamo, Associate Professor, Bergen University College
- Adrian Rutle, PhD-student, Bergen University College
- Zinovy Diskin, Researcher, University of Waterloo, Canada
- Gabriele Taentzer, Professor, University of Marburg, Germany
- ...
Diagram Predicate Framework (DPF)

DPF - Results and applications

- Generalized sketches as institutions, ACCAT’07
- Context and context awareness, ISOLA’08
- Version control, FASE’09
- Strict meta-modeling, TOOLS’09
- Constraint aware model transformations, NWPT’09 + FASE’10
Diagram Predicate Framework (DPF)

DPF - Current projects, ideas, and open questions

- Distributed version control
- Version control based on categorical complements
- Loose meta-modeling
- Extensions of the framework to typed and/or attributed graphs
- Logic of dependencies
- Specification constraints and their logic
- Generalization of Makkai’s construction (pre-sheaf theory)
- Amalgamation in the fibred setting (van-Kampen square)
- Fibred vs. indexed definition of generalized sketches
- Tool design and development
- ...

Uwe Wolter
Generalized Sketches and Indexed vs. Fibred Semantics
Thanks for your attention!