Logics for OO information systems: a semantic study of object orientation from a categorial substructural perspective

de Haas, E.

Citation for published version (APA):
Chapter 5

Methodology: semantics, logic and applications

*Computer science is not only a study of basic theory, and it is not just the business of making things happen. It's actually a study of how things happen.*

*Robin Milner*

What is the semantics of object oriented information systems good for? In this chapter we describe the *methodology* of semantic research underlying the investigations on object orientation in this thesis. Formal semantics is a curious form of mathematics but provides important insights when done properly. The formal semantics also form a valuable basis to research applications of object oriented information models. In this chapter we validate our research on its methodology and point to some gains of the research done so far.

The formal semantic studies on object orientation in this thesis delivers one major artifact: the *logic of categorial graphs*. We use the arsenal of modern logic to provide a proper semantics for the concepts studied. We will place the logic of categorial graphs in its logical context. We give its roots and its logical motivation, and will discuss matters that are important for this kind of logic, focusing on its use in semantical studies.

The theory we developed in this thesis for object oriented information models, of course, gives more then insight only. It provides a basis for research in various application domains that use object oriented information models. One nice application that is a direct application of this research, is the use of the logic for categorial graphs for data mining algorithm analysis ([HaasAdriaans99]). We will elaborate on this application and other possible applications in the second half of this chapter.
5.1 Formal Semantics

It can not be stressed enough that designers, developers and serious users of modeling, database or programming languages need a complete and accurate understanding of the semantics (or intended meaning) of those languages. A rigorous mathematical theory of semantics of a language is needed to support correct description and implementation of its meaning. Furthermore it is indispensable for verification and systematic development of programs written in the language. Of course semantics can be constructed in many different ways. Below we give an overview and describe the approach to semantics for object oriented information systems we have taken in this thesis.

5.1.1 Semantics in computer science

A formal semantics\(^1\) for a language is a mapping from that language to some mathematical structure that models the universe of discourse. In this setting the considered language is called "object language". The mathematical structure is the "semantic domain" and should capture the meaning of the constructs and concepts we want to express in the object language. The elements of the semantic domain have mathematical properties, which exploit the behavior of these elements. In order to do mathematics with the semantic domain we also need a "meta-language" with which we can reason about the semantic domain.

In computer science 'semantics' became an issue when the controversy and confusion over how to interpret the definition of ALGOL-60 started to become embarrassing. Some means had to be found of identifying the meaning of a programming language in a precise and unambiguous way. The, now classical, solution of Scott and Strachey ([ScottStrachey71]) was to associate a mathematical denotation to each phrase of the object language. They discovered Church's λ-notation suitable for this task and constructed a mapping from ALGOL-60 phrases to a λ-language. In this λ-language, the meta-language, one could specify the meaning of the object-language, ALGOL-60. But for precise comprehension this meta-language also needed a semantic definition, which Scott and Strachey provided for in the shape of reflexive domains. Semantics constructed in this manner are generally called denotational semantics.

The similarity between the notion of (denotational) semantics in the programming linguistics as described above, and the notion of semantics in mathematical logic has led to a strong influence of the latter in the field of computer science.

\(^1\)The definition of the concept "semantics" given here is tailored for computer science. We are very well aware that this concept if often used in a much broader setting or in different fields of science.
semantics. In mathematical logic a semantic domain together with a homomorphism from some syntax to that domain is called an 'interpretation'. Given a notion of 'truth' in the domain, an interpretation is a 'model' of a logic if all of the axioms of the logic theory are true in the interpretation and all the inference rules preserve truth. In the case of programming languages we can think of the (denotational) semantics as a model of the language its execution on some (abstract) machine (Floyd-Hoare theory). We only need to explicify some notion of truth in the semantic domain to obtain a proper logic from the interpretation. For example in the case of $\lambda$-calculus models we might use realizability as our notion of truth, in which case all that really needs to be checked is that the conversion rules are valid.

From the above it should be clear that in doing semantics by following the techniques of Scott and Strachey one is concerned with giving mathematical models for programming languages. This is in contrast to the axiomatic approach of other major frameworks of semantics such as Hoare Logic [Cousot90] and Structured Operational Semantics [Milner90] in which the execution behavior of programs is formalized by inventing axioms and rules for the basic programming language primitives. In this way one obtains a 'syntactic' system in which one can deduce programming expressions to be equivalent. Nevertheless one is -in a roundabout way- giving (via the axioms and rules) the linguistic primitives behavior that the interpreting objects should have when we would build a model for the execution\(^2\). It should be clear that the connection between the two kinds of semantics can be tight.

### 5.1.2 Semantics for categorial graphs

In this thesis we designed a 'concrete' discourse model for categorial graphs. This model is abstract enough to have different 'realizations', keeping its core notions intact. By realizations we mean that we can translate the discourse models to actual UML object schemas, real database incarnations, or runs of a program coded in an object oriented programming language.

Because of the appealing abstractness of the semantics domains in the field of logic we constructed the semantics for object oriented information systems in a logical setting. Moreover the logic and the semantics are specifically tailored for the domain of object orientation, and did not choose to code the semantics in a universal language like first order logic with is standard relational semantics. In this regard we take a route that witnesses a relatively new development in logic and theoretical computer science. Universal languages like first order or second order logic become less popular and there is a trend to construct different specific logics for different applications. Let us, for example (or for intimidation), cite

\[^2\text{and would have a 'direct' interpretation function}\]
the famous logician Yuri Gurevich from his manifest of 'logic and the challenge of Computer Science' ([Gurevich88]):

[...]
*But the new applications call, we believe, for new developments in logic proper. First order predicate calculus and its usual generalizations are not sufficient to support the new applications.*

[...]
*It seems that we (the logicians) were somewhat hypnotized by the success of classical systems. We used first-order logic where it fits and where it fits not so well. We went on working on computability without paying adequate attention to feasibility. One seemingly obvious but nevertheless important lesson is that different applications may require formalizations of different kinds. It is necessary to "listen" to the subject in order to come up with the right formalization.*

[...]

So to formalize a specific collection of notions, specific logics are used. Many exotic systems like Horn clause logic, temporal logic, second order polymorphic lambda calculus, dynamic logic, order sorted logic, modal logic, infinitary logic, intuitionistic higher order type theory, continuous algebra, intensional logic and linear logic have been proposed to handle notions of concurrency, time, overloading, exceptions, non-termination, program construction and even natural language. As examples of new application specific logics include various variants of 'linear logic', let us cite some of its ideology ([Girard87]):

[...]
*For logic, computer science is the first real field of application since the applications to general mathematics have been too isolated. The applications have a feedback to the domain of pure logic by stressing neglected points, shedding new light on subjects that one could think of as frozen into desperate staticism, as classical sequent calculus or Heyting's semantics of proofs.*

An example in the tradition of modal logic of application specific modal logics is 'arrow logic' of van Benthem ([Benthem93], [Venema94]) This logic is designed to deal with transitions, and therefore transitions (arrows) are intrinsic objects in the logic. Let us quote some of its motivation([Benthem93]):

*The current interest in logic and information flow has found its technical expression in various systems of what may be called 'dynamic logic' in some broad sense. But unfortunately, existing dynamic logics based on binary transition relations between computational states have high complexity. Therefore, it is worthwhile rethinking the choice of*
5.2. The roots of the logic of categories

A relatively simple dynamic base system forming the 'computational core' that we need, without getting entangled in the complexity engendered by the additional 'mathematics of ordered pairs'.

[...]
This may be seen by developing an alternative, namely a modal logic of 'arrows', which takes transitions seriously as dynamic objects in their own right.

Of course there are enormous differences between these logics mentioned above. The only similarity lies in the fact that these logics are designed to talk 'conveniently' about a specific set of notions, by incorporating these notions on a high level in the formal system.

5.2 The roots of the logic of categories

The logic of categorial graphs that is designed to provide a semantics for object oriented information system languages has been built using the tools of modern mathematical logic. The logic has a very archetypical modal operator in the adjacency operator, but also the aggregation and the empty type could be seen as modal operators. On the other hand the aggregation or composition operator is very well known from substructural logics. Below we take a look at these roots of the logic of categorial graphs.

5.2.1 The categorial graph logic as a modal logic

Intuitively the adjacency modality has its roots in modal logic. The adjacency operator has a property in common with the archetypical modal operators: the adjacency operator gets a meaning only when we know in which world (object) in the model a formula with the adjacency operator is evaluated. To get an idea, we list some basic characteristics of modal logic.

The language of modal logic can be seen as the language of propositional logic to which some operators have been added. Modal languages are interpreted in so called relational structures. Modal logic has been very successful in providing expressive languages and reasoning calculi for various application domains that could be modeled in relational structures. The abstractness (and therefore general) of relational structures makes them very suited for such a task. A relational structure is a nonempty set of items on which a number of relations has been defined. With a modal language one can speak about such relational structures by interpreting the modal operators by the relations. To make things more concrete we give the definitions.
5.2.1. **Definition.** (relational structure) A relational structure is a tuple $F$, whose first component is a non-empty set $U$ we call *universe* or *domain* of $F$, and whose remaining components are *relations* on $U$.

For example a set of objects $U$ together with a relation $R$ that relates two objects when one object is the adjacent of the other, is a relational structure.

5.2.2. **Definition.** (modal language) Let $O$ be a nonempty set of operators $\Diamond$, where each operator has arity $\rho(\Diamond)$ (the pair $(O, \rho)$ is called a *similarity type*), and let $\text{Prop}$ be a set of proposition letters, then the modal language $L$ for $(O, \rho, \text{Prop})$ is defined as follows:

$$L := \text{Prop}|L \sqcap L \sqcup L |\neg L |\Diamond (L_1, \ldots, L_{\rho(\Diamond)})$$

Where $\Diamond$ ranges over the modal operators in $O$ and $L_i$ is an $L$ formula (with an index). For non-nullary modal operators $\Diamond$ we define its dual $\Box$ as follows: Let $A_1, \ldots, A_{\rho(\Diamond)}$ be formulas in $L$ then

$$\Box(A_1, \ldots, A_{\rho(\Diamond)} := \neg \Diamond (\neg A_1, \ldots, \neg A_{\rho(\Diamond)})$$

The archetypal modal language is the language with one unary modal operator $\Diamond$. An example of a modal formula in this archetypal language is $\Diamond P_1 \sqcup \Diamond (P_1 \sqcap \Diamond P_3)$ (where $P_i \in \text{Prop}$).

5.2.3. **Definition.** (model) Let $L$ be a modal language An $L$-frame is a relational structure $F$ with the following ingredients:

1. a non-empty universe $U$,
2. for each modal operator $\Diamond$ of arity $\rho(\Diamond)$ in $L$, a $\rho(\Diamond) + 1$-ary relation $R_\Diamond$.

Given a *pre-valuation* $V : \text{Prop} \mapsto 2^U$ mapping proposition letters to a set of elements in $U$, we can define an $L$-model as a $L$-frame with a pre-valuation:

$$\mathcal{M} = (F, V)$$

In the model we say that $P$ is true in $x$ (or $x$ satisfies $P$) when $x \in V(P)$.

The model for the archetypical modal language with one unary model operator $\Diamond$ will contain a universe $U$, a binary relation $R_\Diamond$ on $U$ and a valuation mapping proposition letters of the modal language to sets of elements in $U$. The relational structure with objects and a relation that relates adjacent objects, together with a valuation would be an appropriate model for this archetypical language.
5.2. The roots of the logic of categories

5.2.4. Definition. (semantics) Let \( L \) be a modal language and let \( M \) be an \( L \)-model as above. Then we define the notion of a formula \( A \) being satisfied in model \( M \) in element \( x \in U \) (notation \( M, x \models A \)) as follows:

\[
\begin{align*}
\mathcal{M}, x \models P & \text{ iff } x \in V(P) \\
\mathcal{M}, x \models \bot & \text{ never} \\
\mathcal{M}, x \models \neg A & \text{ iff not } \mathcal{M}, x \models A \\
\mathcal{M}, x \models A \land B & \text{ iff } \mathcal{M}, x \models A \text{ and } \mathcal{M}, x \models B \\
\mathcal{M}, x \models A \lor B & \text{ iff } \mathcal{M}, x \models A \text{ or } \mathcal{M}, x \models B \\
\mathcal{M}, x \models \lozenge (A_1, \ldots, A_n) & \text{ iff for some } y_1, \ldots, y_n \in U \text{ with } xR_y y_1 \ldots y_n \\
& \quad \text{ we have } \mathcal{M}, y_i \models A_i \ (1 \leq i \leq n) \\
& \quad \text{ for all modalities } \lozenge \text{ of } L
\end{align*}
\]

Now we say that the interpretation of a formula \( A \) in model \( M \) is the set \( \{x | M, x \models A\} \).

Modal logics have axiomatics. The axioms and rules for the modal operators that are interpreted in relations structures all have a small number of axioms and rules in common. For unary modalities these are the axioms and rules for the minimal normal modal logic \( K \). For the modalities with arity greater then 1 similar minimal systems exist.

5.2.5. Definition. (Axiomatics for the minimal modal logic \( K \))

- all axioms and rules of propositional logic for the logic with \( \land, \lor \), and \( \neg \)
- rules for the modalities in the minimal modal logic

\[
\begin{align*}
(\lozenge \text{Distribution}) & \quad \lozenge (A \lor B) \rightarrow (\lozenge A \lor \lozenge B) \\
(\lozenge \text{Necessation}) & \quad \frac{\neg A}{\neg \lozenge A}
\end{align*}
\]

There is also a well known alternative formulation of the rules for \( K \) based on the \( \square \) (the dual modality of \( \lozenge \)) which we sometimes will use.

- rules for the modalities in the minimal modal logic based on \( \square \)

\[
\begin{align*}
(\square \text{Distribution}) & \quad \square (A \rightarrow B) \rightarrow (\square A \rightarrow \square B) \\
(\square \text{Necessation}) & \quad \frac{A}{\square A}
\end{align*}
\]
Modal logics that are complete for relational structures in which the structures have some specific behavior have addition axiomatizations that correspond to the behavior. The behavior of the relations (and the interactions between different relations in a relational structure) is normally defined by constraints. These constraints on the relations have corresponding axioms and rules in the axiomatics\(^3\).

There are numerous examples of logics that talk about complex structures using a modal operator. The tree logic ([BlackburnEtAlii93]) talks about complex tree structures, and arrow logic ([Benthem93]) talks about transition systems.

In the next chapter we will do logical analysis of the categorial graph logic in a pure modal setting. This analysis will both give more insight into the categorial graphs, as well as deliver interesting languages and domains for logical research.

### 5.2.2 The categorial graph logic as a substructural logic

In the categorial graph logic of chapter 4 we have seen an aggregation operator '⋆'. The structural behavior of the domain in which this operator is interpreted is justified in the calculus with which we can reason about the domain by the presence and absence of structural rules for this operator.

In Gentzen style sequential formalisms a substructural logic shows itself by the absence of (some of) the so-called structural rules. Examples of such logics are relevance logic [Dunn86], linear logic [Girard87] and BCK logic [OnoKomori85]. Notable is the substructural behavior of categorial logic\(^4\), which in its prototype form is the Lambek calculus. Categorial logics are motivated by its use as grammar for natural languages. The absence of the structural rules changes the abstraction of sets in the semantic domain to structures, where elements in an aggregation can have position and arity, while in a set they do not.

In figure 5.1 we list the axiomatics of the first order propositional sequent calculus\(^5\), with the axioms, the cut rule, rules for the connectives and the structural rules for exchange, weakening and contraction.

#### 5.2.6. Example. In a domain of sets the following 'expressions' are equivalent, while they are not necessarily so in the domain of structures:

\[
a, a, b, a \approx a, b, b
\]

In a calculus with all the structural rules the features 'position' and 'arity' are irrelevant in the semantic domain, because aggregates that differ in these features

---

\(^{3}\) It is not necessarily so that each constraint corresponds to an axiom or rule. It is more the case that a collection of constraints correspond to a collection of axioms and rules that characterize the set of constraints.

\(^{4}\) See here the inspiration for the name of our basic language building blocks!

\(^{5}\) Note that in the variant we use here we have a special case of the RA rule.
5.2. The roots of the logic of categories

\[(Ax)\] \(A \Rightarrow A\) \hspace{1cm} \[(Cut)\] \(\frac{\Gamma \Rightarrow A, \Delta, \Gamma', A \Rightarrow \Delta'}{\Gamma', \Gamma \Rightarrow \Delta'}\)

\[(L\wedge)\] \(\frac{\Gamma, A, B \Rightarrow \Delta}{\Gamma, A \wedge B \Rightarrow \Delta}\) \hspace{1cm} \[(R\wedge)\] \(\frac{\Gamma \Rightarrow A, \Delta, \Gamma' \Rightarrow B, \Delta}{\Gamma, \Gamma' \Rightarrow A \wedge B, \Delta}\)

\[(L\vee)\] \(\frac{\Gamma, A \Rightarrow \Delta, \Gamma, B \Rightarrow \Delta}{\Gamma, A \vee B \Rightarrow \Delta}\) \hspace{1cm} \[(R\vee)\] \(\frac{\Gamma \Rightarrow A, \Delta, \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \vee B, \Delta}\)

\[(Ex)\] \(\frac{\Gamma, A \wedge B, \Gamma' \Rightarrow \Delta}{\Gamma, B \wedge A, \Gamma' \Rightarrow \Delta}\)

\[(Weak)\] \(\frac{\Gamma \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta}\)

\[(Contr)\] \(\frac{\Gamma, A, A \Rightarrow \Delta}{\Gamma, A \Rightarrow \Delta}\)

\[\text{Figure 5.1: First order propositional sequent calculus}\]

can be proved equivalent with the structural rules. To see this, observe that the left side of the above equation can be transformed to the right side by performing the following operation:

\[a, a, b, a\]

contract \(a, a\) in first two positions to \(a\)

\[a, b, a\]

exchange \(b, a\) in last to positions to \(a, b\)

\[a, a, b\]

contract again \(a, a\) in first two positions to \(a\)

\[a, b\]

weaken expression \(b\) in last position to \(b, b\)

\[a, b, b\]

In the logical analysis in the next chapter we will also analyze the categorial graph logic from a purely substructural logic point of view. This will give the logic of categorial graph a 'more explored' foundation, and will enable us to list some interesting characteristics of fragments of the categorial graph logic. We will also see that there is a natural transition from the modal way of looking at categorial graphs and the substructural way. This will become clear in chapter 6.
5.2.3 Related logics of information systems

The logic of categorial graphs is not the only approach to formalize the world of object oriented information systems using logic. Two influential proposals are 'O-Logic' of Maier ([Maier86], extended in [KiferWu93]) and 'F-Logic' of Kifer and Lausen ([KiferLausen89]). In these proposals the notions of object orientation are embedded in first order (predicatc!l) logic (O-Logic 'contains' first order predicate logic), and even higher order logic (F-Logic). The strong languages enables one to express all possible features easily and elegantly, and can be used to reason in these strong systems about the features expressed. However, because of the wealth of expressive power, these systems are less suited for logical analysis of the features themselves. Moreover the reasoning task is computationally undecidable, because the systems of first order predicate logic and higher order logic are computationally undecidable.

There are also related systems of Information logics, that were inspired from a more general (not particularly object oriented) conception of information and knowledge. Examples of these logics are 'description logics' ([Baader96], [Areces00]), 'feature logics' ([Rounds97]), logic of Information flow ([BarwiseSeligman97]).

The approach we take to object oriented information system logics distinguishes itself from the other proposals by the fact that we use thoroughly the achievements of modal and substructural logic. This enables us to analyze the notions from a perspective that is 'lower' in generality than most other approaches to object orientation, that are first order and higher order theories. Our approach enables us to analyze the notions of object orientation without carrying the burden of the generality of first order logic, which includes over-expressiveness (one can say much more in an over-expressive language than actually needed to capture the notions) and computational intractability (first order theories are in general undecidable). In our approach we can analyze the notions more directly (in a language tailored very closely to the notions) and we use theories with computationally nicer behavior. We will see some nice achievements from this approach in the next chapter.

In a logical sense our approach relates more closely to the mentioned information logics. These logics are tailored to reason about information (although not necessarily object oriented). It is therefore an interesting exercise to map the features we came up with to the frameworks of these information logics. This exercise is subject to future work.
5.3 Applications

Next to applications in the practice of object oriented information systems and logic, there are interesting applications for the logic of categorial graphs in fields that need a good understanding about the structure of object oriented information. One such field that really needs insight in information structures and that has both practical and theoretic involvement is data mining.

5.3.1 Applications in object oriented information system practice

The most direct applications can be found in the field of model checking of the large and complex information models that information system designers construct in practice. Modern software development methodologies (e.g. UP, see chapter 1) invite designers to specify parts of the systems in small pieces (components) from many different views (in different use cases, and in different level of abstractness -requirements, analysis, design and implementation). The scattered model should be checked on consistency and validity (for example in the sense that it should be satisfiable). Automatic support for this task is still very rudimentary and far from complete for object oriented models. A system like the logic of categorial graphs give a theory to handle the checking of the more involved constraints that are formulated for the information models.

5.3.2 New computational applications

The 'tailored' logical approach to object oriented information taken here suggests new ways of looking at other tasks that use object oriented information structures and benefit from logical systems. An example of a new task is the use of inductive logic for data mining purposes. In data mining one processes complex structured data (often modeled in OO languages like UML) and benefits from inductive reasoning models; i.e. from logic.

In [HaasAdriaans99] we proposed a framework for data mining algorithms based on the logic of categorial graphs. The research was initiated by observing interesting connections between data mining, inductive logic programming and grammar induction ([AdriaansHaas99]). The logic of categorial graphs facilitates the design of efficient Data mining algorithms using techniques from the field of inductive logic programming. This framework is based on the following idea:

The processing of more complex structured data calls for new algorithms. The inductive algorithms used for data mining on tabular (relational) structures are theoretically impaired with the computational complexity of full first order predicate logic. This is a problematic issue if one designs algorithms for induction on more complex structures than flat relations. This observation suggests that we need to found the structures on which we do data mining on a system with much
better computational properties. This complies with the broader trend in logic in computer science that suggests that one needs to find logics tailored more closely to the application domain than a general language as full first order predicate logic.

Complex data in modern information technology practice is often written down in the industry standard language UML. A logic that effectively bears the main features of object oriented languages like UML seems ideal for the task to support the design of inductive algorithms for finding patterns, because it has both

1. the ability to express complex patterns in complex data structures,

2. and a calculus that enables one to perform correct induction steps on discovered patterns.

A strongly related new task that could benefit from our approach is the task of learning from a logical perspective. There is a strong relationship between a learning strategy, its formal learning framework and its logical representational theory. This relationship enables one to translate learnability results from one theory to another. Moreover if we go from a classical logic theory to a more specialized logic like the logic of categorial graphs or a substructural logic theory, we can transform learnability results of logical concepts to results for information models or string languages. In [AdriaansHaas00] we demonstrated such a translation by transforming the Valiant learnability result for Boolean concepts to a learnability result for a class of string pattern languages.

### 5.3.3 Logical and philosophical repercussions

In the light of the logical context the system for categorial graphs also puts forward some interesting questions for logical research. These questions will be asked in the next chapter. Furthermore the clear-cut analysis of the domain, involving 'objects' and 'partial descriptions', have surprisingly tight connections to issues in philosophy. When one realizes that information system modeling is a task of accurately and precisely modeling a part of reality, using basic notions like 'object', 'property' and 'description', applications in philosophy are certainly possible. A formal theory of information can provide a way to explicity issues in philosophical considerations or debate. We will show such an application in chapter 7.

---

6 Data mining is learning patterns from data.
5.4. Summary

In this chapter we sketched the research domains that we are touching in the analysis of object orientation. We have mentioned research on computer systems semantics and mathematical logic. In the following chapter we will analyze the logic of categorial graphs exploiting the logical roots that were pointed out here.