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 6

Logic of object oriented information

1.1 Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.
2.034 Die Struktur der Tatsache besteht aus den Strukturen der Sachverhalte
2.04 Die Gesamtheit der bestehenden Sachverhalte ist die Welt

(Ludwig Wittgenstein, Tractatus logico-philosophicus)

The informal practice of object-oriented information systems has been analyzed in depth in Chapters 3 and 4. This culminated in our eventual proposal for precise mathematical 'discourse models'. These came with two languages streamlining the usual object-oriented discourse: a mixed graphical-symbolic one of categorial graphs and its more traditional 'meta-language', both reflecting the key features of adjacency, parts and products. Moreover, we gave a substructural calculus of resource-conscious proof rules that fits informal reasoning about specified constraints on objects and their properties. This package of formal languages, semantic structures, and proof calculi may be viewed on its own as our proposal for a 'precisified practice' in the area. But also, in Chapter 5, we pointed out how such a model can serve as a basis for new tasks, not yet studied systematically in the literature, such as *learning* resource-sensitive object-oriented structures. In this sixth chapter, however, we want to use our model in another mode, namely, for analyzing the logical properties of object oriented programming practice. In particular, we will show how the framework that we have developed fits with a number of independent developments in categorial and modal logic. For this purpose, we develop a new modal 'streamlined version' of the earlier systems which facilitates the comparison. This correspondence allows us to draw some precise conclusions about expressiveness, axiomatization, and complexity of the object-oriented paradigm. But also conversely, the resulting system presents some interesting challenges to modal and categorial logicians, as we shall point out in due course.
Chapter 6. Logic of object oriented information

As usual, any logical analysis involves two mathematical decisions, which we repeat at the outset:

1. Which *structures* do we use to capture the items from the real world that are at stake?
2. Which *languages* do we use to talk about these items?

The following two sections state our answers, being a recapitulation of what we did in the preceding chapters, but with some new twists.

6.1 Models for object-oriented information systems

Chapters 3 and 4 gave a semi-formal, but rigorous, description of practically plausible object-oriented models and their basic structure. We now sharpen this up in two ways:

- define an *intended model*, which is a fixation of the exact structure we would like to study; the archetype of the concrete models of chapter 4.
- define *abstract models*, which are abstract versions of the former, providing greater generality and easier access for logical theorizing.

6.1.1 Intended models

The 'discourse models' of chapter 4 were meant to directly describe the semantic intuitions behind object oriented information systems. Now we give a *concrete* definition for logical working purposes.

6.1.1. Definition. (Intended model) There are two components:

1. One domain with a set of 'whole objects' $E$, and products of whole objects ('aggregates') in set, multiset, and list flavors.

2. Another family of domains $\{A_e | e \in E\}$, consisting of 'partial descriptions' and their products.

These two domains are related as follows:

1. a 'structure adjacency relation' $f_R$ relates whole objects to their partial descriptions

\footnote{In the concrete model, the partial descriptions are precisely the substructures of the total adjacency structure.}
6.1. Models for object-oriented information systems

2. The 'extension relation' \textup{ext} relates a partial description to other 'more informative' partial descriptions, and whole objects.

More precisely, an intended model gives us the following:

1. **Whole objects:** Atomic objects $E^\text{At}$ and all the sets, multisets, and lists of atomic objects (seen as various sorts of aggregate objects) together with the appropriate 'aggregation' operations $\ast$ (set union\textsuperscript{2}, multiset union and list concatenation), and the 'substructure' relations $\leq$ (subset\textsuperscript{3}, submultiset, sublist) for the different flavors. Thus we allow all three options at the same time, in one multi-sorted domain of objects with partially defined product operations. This domain of whole objects is denoted by $E$.

2. **Partial descriptions:** For each $e \in E$, we have a domain of partial descriptions $A_e = e \times E$ (the 'labeled object domain'), again with its aggregation operations\textsuperscript{4} $\ast_e$ and substructure relations\textsuperscript{5} $\leq_e$.

3. **Adjacency mapping:** A function $f_R : E \mapsto \bigcup_{e \in E} A_e$ such that $f_R(e) \in A_e$, which maps each whole object to its largest partial description.

4. **Extendibility relation:** $\text{Ext} : E \cup \bigcup_{e \in E} A_e \mapsto \mathcal{P}(E \cup \bigcup_{e \in E} A_e)$ where $\text{Ext}(e) = \{e\}$ for all $e \in E$ and $\text{Ext}(a) = \{b | a \leq_e b \leq_e f_R(e)\} \cup \{e\}$ for all $a \in A_e$ where $a \leq_e f_R(e)$.

\[ \triangle \]

An attentive reader may have noted that we left out the empty object '1' and the empty descriptions '(e, 1)' that were present in the models for categorial graphs in the original analysis in chapter 4. These empty entities were introduced in the algebraic semantics for categorial graphs for convenience. It enabled us to take monoids as basic structures, and therefore we were able to elegantly express the algebraic properties. As we mentioned at its introduction in chapter 3, the empty edge is not a necessary artifact, even though we could give it a proper meaning. The models in this chapter have a more minimal nature, to enable the analysis of the object oriented structures. We could introduce the empty edge here without a problem, but then we will need some additional rules and axioms to enforce its behavior, while we actually are not very interested in this artifact for the study of object orientation. An artifact that is convenient in one setting (the algebraic) may prove to be a nuisance in another setting (the logical).

\textsuperscript{2}In our notation, set union will be $\ast_{\text{set}}$, multiset union $\ast_{\text{multiset}}$ and list concatenation $\ast_{\text{list}}$. Using the $\ast$ by itself, we abstract over the choice of structure type.

\textsuperscript{3}As notation then, subset will be $\leq_{\text{set}}$, submultiset $\leq_{\text{multiset}}$ and sublist $\leq_{\text{list}}$. By using $\leq$ we abstract over the choice of structure type.

\textsuperscript{4}Again, we actually have several: $\ast_{\text{set}}, \ast_{\text{multiset}}, \ast_{\text{list}}$.

\textsuperscript{5}And once more, we have several: $\leq_{\text{set}}, \leq_{\text{multiset}}, \leq_{\text{list}}$.\
6.1.2 Abstract models

Next, we take a more abstract viewpoint, highlighting what we see as the essential features of the preceding models. In particular, these are: the use of two kinds of entities: whole objects and partial descriptions, each with their own product operations, but living in harmony through suitable 'connection relations'. Getting a bit ahead of things, what follows are typical relational models for modal logic, satisfying some suitable constraint:

6.1.2. DEFINITION. (Abstract model)

1. First, we have a domain $U$

2. Then there are two families of ternary relations $Q^E$, $Q^A$, where each family consists of three relations $Q^{setX}$, $Q^{multisetX}$, $Q^{listX}$. When we abstract over the precise choice of $Q^X$, we sometimes write $Q^{structX}$.

3. Finally, we have binary relations $R_1$, $R_2$, and $S$.

In compact form an abstract model $\mathcal{M}$ is written as follows:

$$\mathcal{M} := < U, Q^{setE}, Q^{setA}, Q^{multisetE}, Q^{multisetA}, Q^{listE}, Q^{listA}, R_1, R_2, S >$$

$\triangle$
6.1. Models for object-oriented information systems

![Figure 6.2: Topology of the abstract model](attachment:fig6_2.png)

The motivating interpretation for this similarity type is the following. We have objects and descriptions living together in the total domain, each with ternary 'composition relations' of the form 'aQbc': \( a \) is a composition (of one of our various sorts) of \( b \) and \( c \), in that order. This ternary style makes sure that the product operations are not necessarily total (relieving us of the duty to interpret every weird 'hybrid product') while also remaining uncommitted on the issue of whether forming aggregates is a single-valued partial function, rather than a multi-valued one. Moreover, 'communication' is arranged as follows. Objects are related to their descriptions by the relation \( R_1 \), while in the opposite direction, \( R_2 \) takes descriptions back to the objects figuring in them, either as the 'leading object' or as another one involved in the relevant property of the former. Finally, \( S \) is the extension relation, which is characteristic for (after all) partial descriptions, but allowing 'culmination' in the 'main object' of the given description.

As usual, the move toward abstract models is not just a trick to make the theoretician's life easier. The above format may also suggest new applications, and new ways of looking at 'object oriented information systems'. In this chapter, we gain two things by working this way:

1. a clear view of the semantics for categorical graphs,
2. transfer of results from (and to) established modal logic.

6.1.3 Representation

What is needed to represent an abstract model as a concrete one? On these abstract models we will formulate constraints such that the abstract models have (much of) the same characteristics as a concrete model. In the abstract models the elements and relations are abstract, but 'secretly' we imagine them to have some intended meaning. The domain \( U \) contains, in our intended meaning, both
the whole objects and the partial descriptions. When enough structure is given in the abstract model we will be able to distinguish whole objects from partial descriptions. In the abstract view, when no characterizations are given, we just say our abstract domain contains elements. At the start of our analysis all the relations are also without any restrictions, but we will further on constrain relations in $Q^E$ to model the aggregation (product) on whole objects (separate relations for sets, multisets, and lists) and the relations in $Q^A$ to model aggregation on partial descriptions (also separate relations for sets, multisets, and lists). Furthermore $R_1$ will model adjacency relating whole objects to its partial descriptions; $R_2$ will model the relation between a partial object and the whole object that takes part in the adjacency. Finally $S$ will model extendibility. In the coming sections we will encounter the constraints that make all this happen.

### 6.2 Modal languages

To talk about the objects, the partial descriptions, and their interactions, we start with a modal language. The modal language is a slight extension to the meta language of categorial graphs in three ways:

- the *adjacency* modality is split up in *two* modalities, the first relating an object to its partial description, and the second relating the partial description to the actual adjacent object

- the *aggregation* modality will be accompanied by two additional modalities that look at the aggregation from the two other possible perspectives. More-
over we will have separate aggregation modalities for the different structures
'lists', 'multisets', and 'sets' and the different entities 'objects' and 'partial
descriptions'.

- The notion of *extendibility* in the model will get a modality in the language
to enable us to talk about it.

In the modal language we will omit the constant \(1\) for the empty object, for the
reason we explained above.

The relations of the abstract model from above are now interpreted by the
following modalities:

- unary \(\Diamond_1, \Diamond_2, \Diamond_3\) modalities for the adjacency relations
- unary \(\circ, \circ^U\) modalities for the extendibility relation
- binary modalities \(\ast_{\text{list}E}, \ast_{\text{multiset}E}, \ast_{\text{set}E}, \ast_{\text{list}A}, \ast_{\text{multiset}A}, \ast_{\text{set}A}\)
  for the aggregations (products) on whole objects
  (for all types of structures three for the 'triple view' of a ternary relation)
- binary modalities \(\ast_{\text{list}A}, \ast_{\text{multiset}A}, \ast_{\text{set}A}\)
  for the aggregations (products) on partial descriptions
  (for all types of structures three for the 'triple view' of a ternary relation)

The modalities will be interpreted by the relations in the abstract model. The
modalities embedded in a modal language will enable us to talk about complex
objects, precisely like we did with the meta language for categorial graphs.

### 6.2.1 Definition language and semantics

#### 6.2.1.1 Definition. (Modal language of categorial graphs)
Let \(\text{Prop}\) be a set of propositional variables. Then we define the *modal language of categorial graphs* \(L\) as follows:

- **Boolean:** \(L = \text{Prop} | L \sqcup L \sqcup L \rightarrow L\)
- **Adjacency:** \(\Diamond_1 L | \Diamond_2 L | \bigcirc_1 L | \bigcirc_2 L | \Diamond_1 L \sqcup \Diamond_2 L \sqcup \bigcirc_2 L \sqcup L\)
- **Extension:** \(\circ L | \circ^U L\)
- **Aggregate on objects:** \(L = L \ast_{\text{struct}E} L \sqcup L \ast_{\text{set}E} L \sqcup L \ast_{\text{list}E} L\) where \(\text{struct} \in \{\text{set, multiset, list}\}\)
- **Aggregate on partial descriptions:** \(L = L \ast_{\text{struct}A} L \sqcup L \ast_{\text{set}A} L \sqcup L \ast_{\text{list}A} L\)
  where \(\text{struct} \in \{\text{set, multiset, list}\}\)

The modal language of categorial graphs is interpreted in the abstract model
as follows
DEFINITION. (Semantics) Let \( V: \text{Prop} \rightarrow 2^U \) be a valuation that assigns subsets of \( U \) to atomic symbols. Let \( P \in \text{Prop} \) and \( A, B \in L \). Furthermore let \( s, t, u \in U \). Then abstract model \( \mathcal{M} \) interprets \( L \) as follows:

- **Boolean** (standard as in section 5.2, but given here for completeness):
  \[
  \mathcal{M}, s \models P \quad \text{iff} \quad s \in V(P) \\
  \mathcal{M}, s \models A \cap B \quad \text{iff} \quad \mathcal{M}, s \models A \text{ and } \mathcal{M}, s \models B \\
  \mathcal{M}, s \models A \cup B \quad \text{iff} \quad \mathcal{M}, s \models A \text{ or } \mathcal{M}, s \models B \\
  \mathcal{M}, s \models \neg A \quad \text{iff} \quad \mathcal{M}, s \not\models A \\
  \]

- **Adjacency** (interpreted by \( R_1 \) and \( R_2 \)):
  \[
  \mathcal{M}, s \models \Diamond_1 A \quad \text{iff} \quad \exists t(sR_1 t \text{ and } \mathcal{M}, t \models A) \\
  \mathcal{M}, s \models \Diamond_2' A \quad \text{iff} \quad \exists t(tR_1 s \text{ and } \mathcal{M}, t \models A) \\
  \mathcal{M}, s \models \Box_1 A \quad \text{iff} \quad \mathcal{M}, s \not\models \neg \Diamond_1 \neg A \\
  \mathcal{M}, s \models \Box_2' A \quad \text{iff} \quad \mathcal{M}, s \not\models \neg \Diamond_2' \neg A \\
  \mathcal{M}, s \models \bigcirc_1 A \quad \text{iff} \quad \exists t(sR_2 t \text{ and } \mathcal{M}, t \models A) \\
  \mathcal{M}, s \models \bigcirc_2' A \quad \text{iff} \quad \exists t(tR_2 s \text{ and } \mathcal{M}, t \models A) \\
  \mathcal{M}, s \models \bigtriangledown_1 A \quad \text{iff} \quad \mathcal{M}, s \not\models \neg \bigcirc_1 \neg A \\
  \mathcal{M}, s \models \bigtriangledown_2' A \quad \text{iff} \quad \mathcal{M}, s \not\models \neg \bigcirc_2' \neg A \\
  \]

- **Extension** (interpreted by \( S \)):
  \[
  \mathcal{M}, s \models oA \quad \text{iff} \quad \exists t(sSt \text{ and } \mathcal{M}, t \models A) \\
  \mathcal{M}, s \models o'^{A} \quad \text{iff} \quad \exists t(tSs \text{ and } \mathcal{M}, t \models A) \\
  \]

- **Aggregate on objects** (interpreted by the relations in \( Q^E \)):
  \[
  \mathcal{M}, s \models A \ast^1_{\text{struct} E} B \quad \text{iff} \quad \exists u(sQ^\text{struct} E tu \text{ and } \mathcal{M}, t \models A \text{ and } \mathcal{M}, u \models B) \\
  \mathcal{M}, s \models A \ast^2_{\text{struct} E} B \quad \text{iff} \quad \exists u(tQ^\text{struct} E su \text{ and } \mathcal{M}, t \models A \text{ and } \mathcal{M}, u \models B) \\
  \mathcal{M}, s \models A \ast^3_{\text{struct} E} B \quad \text{iff} \quad \exists u(tQ^\text{struct} E us \text{ and } \mathcal{M}, t \models A \text{ and } \mathcal{M}, u \models B) \\
  \]

- **Aggregate on partial descriptions** (interpreted by the relations in \( Q^A \)):
  \[
  \mathcal{M}, s \models A \ast^1_{\text{struct} A} B \quad \text{iff} \quad \exists u(sQ^\text{struct} A tu \text{ and } \mathcal{M}, t \models A \text{ and } \mathcal{M}, u \models B) \\
  \mathcal{M}, s \models A \ast^2_{\text{struct} A} B \quad \text{iff} \quad \exists u(tQ^\text{struct} A su \text{ and } \mathcal{M}, t \models A \text{ and } \mathcal{M}, u \models B) \\
  \mathcal{M}, s \models A \ast^3_{\text{struct} A} B \quad \text{iff} \quad \exists u(tQ^\text{struct} A us \text{ and } \mathcal{M}, t \models A \text{ and } \mathcal{M}, u \models B) \\
  \]

6.2.3. **Example.** Recall the running example of chapter 3 (example 3.1.1) and its formulation in the meta language for categorial graphs of example 4.2.2. As a brief illustration, we show how to express the basic graph expressions in the modal logic for categorial:

\[
\begin{align*}
\text{pilot} & \Rightarrow \Diamond_1 \Diamond_2 \text{name } \ast_1 \Diamond_1 \Diamond_2 \text{empno } \ast_1 \Diamond_1 \Diamond_2 \text{qualif} \\
\text{pilot} & \Rightarrow \Diamond_1 \Diamond_2 \text{roster}
\end{align*}
\]
6.2. Modal languages

In the sections below we will first define a logic that handles adjacency, a logic that handles extendibility, and a logic that handles aggregates, before we present a combined system that can talk about the combination of all those things. Such a logic consists of a language (a fragment of the language from above), a model (a part of the abstract model from above), an interpretation of the logical connectives and modalities, and a collection of axioms and rules that characterize certain constraints on the relations of the abstract model.

6.2.2 Adjacency logics

In the logic of adjacents we focus on a very important feature of a language for object oriented information systems: the ability to construct complex objects using the adjacency operation. In the logic of adjacents we analyze two types of interactions between whole objects and partial descriptions. Firstly we will be able to say things about having adjacents; i.e. we need to relate an object to the partial description that witnesses that this object has some other object as an adjacent. Secondly we need means to say that an object is an adjacent of some other object; i.e. when an object takes part in a partial description.

The two matters of adjacency are expressed using two unary modalities $\diamond_1$ and $\diamond_2$, and are interpreted by the relations $R_1$ and $R_2$ in the abstract model. The $R_1$ relation here relates a whole object to the partial descriptions that describe the fact of having some adjacents. The $R_2$ relation relates a partial description to the object that is the adjacent. In other words, $R_2$ intends to express that a partial description involves some object. This is a direct concept of the object oriented paradigm we have already seen in chapters 3 and 4 that says that all the entities can be looked at as objects in their own right.

Note that here we directly look at the interpretation of an adjacent to its partial description, and do not model the adjacency via an adjacency relation that relates whole objects to whole objects. The need for considering partial descriptions was discovered when formulating the concrete models in chapter 4.

For example, consider object $a$ and partial description of $a$ called $b$ that is the witness of $a$ having as adjacent object $c$. Then $aR_1b$ relates the $a$ object to its $b$ partial description and $bR_2c$ relates partial description $b$ to the object $c$ that is the content of the partial description of $a$.

There is also another possible view on the adjacency relations, the inverse of the relations $R_1$ and $R_2$. This view also provides some nice means of expression. One could say interesting things about the partial descriptions; for example 'the partial description $b$ partially describes object $a$'.

Note that we cannot choose $R_1$ and $R_2$ arbitrarily. The relation modeling adjacency should satisfy some constraints to comply with the intended meaning\(^6\). These constraints are as follows:

1. An object can have a partial description, but a partial description in turn, does not have a partial description anymore

2. A partial description is a witness to an object being an adjacent. An object itself, however cannot be such a witness

3. Sets of objects and partial descriptions are disjoint

4. The universe contains no other entities than partial descriptions and objects with partial descriptions

5. A partial description is always a partial description of an object

In other words the relations, $R_1$ and $R_2$ are at a maximum one step deep; the collections of partial descriptions and objects are disjoint and exhaustive, and all partial descriptions must be related to whole objects by both $R_1$ and $R_2$; i.e.

6.2.4. DEFINITION. (Constraints for the adjacency logic)

\[1. \forall a (\exists b aR_1b \rightarrow \neg \exists c bR_1c)\]

\[2. \forall a (\exists b aR_2b \rightarrow \neg \exists c bR_2c)\]

\[3. \forall a \exists b, c (aR_1b \text{ and } aR_2c)\]

\[4. \forall a \exists b, c (aR_1b \text{ or } aR_2c)\]

\[5. \forall a (\exists b aR_2b \rightarrow \exists c cR_1a)\]

\[\Delta\]

Note that having an adjacent (i.e. being the left side of an $R_1$ relation) is characteristic for a whole object, and being a witness of an adjacent (i.e. being the left side of an $R_2$ relation) is characteristic for a description. Constraint 4 says that all objects have adjacents (remember the possibility for introducing an empty description!) and that all descriptions are witnesses to an object being an adjacent (remember the possibility for introducing empty object!).

The language for the adjacency logic becomes the following:

---

Note that if we would construct a model with a domain with only whole objects and a relation that relates whole object to the whole objects that take part in the partial description. Then we would not have any necessary constraint on the single adjacency relation $R$ and would come up with the modal logic $K$. We could, though, optionally put some constraints like reflexivity or acyclicity or foundedness on $R$, with their well known rules in the logic. We would however not be able to express things involving the partial descriptions like the structure of the adjacency.
6.2.6.2. Modal languages

Figure 6.4: object $a$ with partial description $b$ witnessing that object $c$ is actually adjacent to $a$.

6.2.5. Definition. (Language for the adjacency logic)

\[ L_I := \text{Prop}|L_I \cap L_I|\neg L_I|\Diamond_1 L_I|\Diamond_2 L_I|\Diamond_1^\square L_I|\Diamond_2^\square L_I|\Box_1 L_I|\Box_2 L_I|\Box_1^\square L_I|\Box_2^\square L_I \]

We can now interpret the adjacency modalities, their inverse modalities and all their dual modalities in the abstract model as indicated:

6.2.6. Definition. (Semantics for the adjacency logic)

\[
\begin{align*}
M, s \models \Diamond_1 A & \text{ iff } \exists t(sR_1 t & M, t \models A) \\
M, s \models \Diamond_2 A & \text{ iff } \exists t(sR_2 t & M, t \models A) \\
M, s \models \Diamond_1^\square A & \text{ iff } \exists t(tR_1 s & M, t \models A) \\
M, s \models \Diamond_2^\square A & \text{ iff } \exists t(tR_2 s & M, t \models A)
\end{align*}
\]

As in standard modal logic, we introduce the dual operator of the $\Diamond_I$ by setting $\Box_I := \neg \Diamond_I = \neg$. \[\Delta\]

The modal logic for adjacency will contain at least the axioms and rules of the minimal modal logic $K$ added with the principles that are needed to force the inverse modalities to be interpreted by the inverse relation. The most interesting principles come from the constraints we put on the relations $R_1$ and $R_2$ in the abstract model to force behavior that is similar to that of the concrete models.

6.2.7. Definition. (Axiomatics for the adjacency logic) First we construct the axioms and rules for the minimal logic for the adjacency modalities:

- all axioms and rules of propositional logic for the logic with $\cap$, $\cup$ and $\neg$
Chapter 6. Logic of object oriented information

- rules for the modalities in the minimal modal logic

\[
(\Diamond_1 \text{Distribution}) \quad \Diamond_1 (A \cup B) \rightarrow (\Diamond_1 A \cup \Diamond_1 B)
\]
\[
(\Diamond_2 \text{Distribution}) \quad \Diamond_2 (A \cup B) \rightarrow (\Diamond_2 A \cup \Diamond_2 B)
\]
\[
(\Diamond_1^U \text{Distribution}) \quad \Diamond_1^U (A \cup B) \rightarrow (\Diamond_1^U A \cup \Diamond_1^U B)
\]
\[
(\Diamond_2^U \text{Distribution}) \quad \Diamond_2^U (A \cup B) \rightarrow (\Diamond_2^U A \cup \Diamond_2^U B)
\]
\[
(\Box_1 \text{Necessation}) \quad \neg A \rightarrow \neg \Diamond_1 A
\]
\[
(\Box_2 \text{Necessation}) \quad \neg A \rightarrow \neg \Diamond_2 A
\]
\[
(\Box_1^U \text{Necessation}) \quad \neg A \rightarrow \neg \Diamond_1^U A
\]
\[
(\Box_2^U \text{Necessation}) \quad \neg A \rightarrow \neg \Diamond_2^U A
\]

- then we need to add the rules to relate the modalities to their inverse modalities:

\[
(\Box_1 \text{Inverse}) \quad A \rightarrow \Box_1 \Diamond_2 A
\]
\[
(\Box_2 \text{Inverse}) \quad A \rightarrow \Box_2 \Diamond_2 A
\]

- finally we need to add rules and axioms that reflect constraints on the adjacency relations. In the adjacency logic this amounts to the following principles:

\[
(\Box_1 \text{Step}) \quad \Box_1 \Box_1 \bot
\]
\[
(\Box_2 \text{Step}) \quad \Box_2 \Box_2 \bot
\]
\[
(\text{Disjoint}) \quad \neg (\Diamond_1 T \cap \Diamond_2 T)
\]
\[
(\text{Exhaustive}) \quad \Diamond_1 T \cup \Diamond_2 T
\]
\[
(\text{Description Requires Object}) \quad \Diamond_2 T \rightarrow \Diamond_1^U \bot
\]

The adjacency logic enables us to say things about the domain that have to do with adjacency. Now we have appropriate means to express whether an object is a whole object, or whether an object is a partial description:

- an entity \( x \) is a whole object iff \( x \) satisfies \( \Box_2 \bot \)
- an entity \( x \) is a whole object iff \( x \) satisfies \( \Diamond_1 T \)
- an entity \( x \) is a partial description iff \( x \) satisfies \( \Box_1 \bot \)
- an entity \( x \) is a partial description iff \( x \) satisfies \( \Diamond_2 T \)
- an entity \( x \) is a partial description iff \( x \) satisfies \( \Diamond_1^U T \)
- an entity \( x \) is a partial description of an \( A \) - object iff \( x \) satisfies \( \Diamond_1^U A \)
Note that when we want to connect an object to another object that takes part in its adjacency structure, like in the original definition of the categorial graph meta language, we can simply traverse via a partial description. Suppose we want to say that an object $x$ has an $A$-adjacent object. We expressed this in original language by $\Diamond A$. In this more refined setting we will say $\Diamond_1 \Diamond_2 A$. If we want to express constraints directly on the object adjacency relation (i.e. treat it as a normal relation) we can formulate the corresponding principles using the composed $\Diamond_1 \Diamond_2$ modality.

There are also some nice things to say from a modal logic perspective about the axiomatics of the adjacency logic. There are nice principles deducible from the system. For example: *it is necessary for an object to have an adjacent object via a partial description:*

$$
\frac{
\Diamond_1 T \cup \Diamond_2 T \quad \text{(Exhaustive)}
}{
\Box_1 (\Diamond_1 T \cup \Diamond_2 T) \quad \text{(\Box_1 \text{Necessation})}
\}
$$

$$
\frac{
\Box_1 (\neg \Diamond_1 T \rightarrow \Diamond_2 T) \quad \text{(\Box_1 \text{Distribution})}
}{
\Box_1 \neg \Diamond_1 T \rightarrow \Box_1 \Diamond_2 T
\}
$$

$$
\frac{
\Box_1 \bot \quad (\Diamond_1 \text{Step})
}{
\Box_1 \Diamond_1 \bot \rightarrow \Box_1 \Diamond_2 T \quad \text{modus ponens)
\}
$$

Another observation is that there is only one principle that introduces an asymmetry between the objects and the partial descriptions. This principle is the Description Requires Object axiom. This axiom requires that each complete chain of objects and partial descriptions connected alternatively by $R_1$ and $R_2$ relations always start with an object. I.e.:

$$
\circ \rightarrow_1 \circ \rightarrow_2 \circ \rightarrow_1 \circ \rightarrow_2 \cdots
$$

is an admissible structure in a model for an adjacency structure, but the following is not:

$$
\circ \rightarrow_2 \circ \rightarrow_1 \circ \rightarrow_2 \circ \rightarrow_1 \cdots
$$

An other interesting observation is that we have taken a positive existence property in axioms 'Disjoint' and 'Exhaustive' to characterize that some entity in the model is an object or a description; namely respectively $\Diamond_1 T$ and $\Diamond_2 T$. An interesting alternative is to take the negative existence properties $\Box_2 \bot$ and $\Box_1 \bot$. The axioms then become:

$$(\text{Disjoint}) \quad \neg (\Box_2 \bot \cap \Box_1 \bot)
$$

$$(\text{Exhaustive}) \quad \Box_2 \bot \cup \Box_1 \bot
$$

In the negative formulation we allow entities in the model that are not connected through $R_1$ or $R_2$. Such entities can be either an object or partial description (it actually does not matter which). A hybrid formulation is also a plausible
alternative; for example $\Box_2\bot$ to characterize objects and $\Diamond_2\top$ to characterize descriptions. In a more general setting this matter will be touched in the section on 'further logical considerations' below.

6.2.3 Extendibility logics

In the extendibility logic we isolated the extendibility operation that can talk about extending partial descriptions and objects to more (or equal) informative descriptions or (in the limit) the object itself.

With the categorial graphs of chapter 3 we defined a language that enables one to talk about object oriented information systems from the perspective objects. The analysis for the interpretation in chapter 4 motivated us to introduce partial descriptions and the extendibility relation. The reason that the extendibility found its way in the concrete model is because it models a natural intuition. Coming to the conclusion that these are important features of models for object oriented information systems, it is only fair that we provide constructs for talking from the perspective of partial descriptions as well.

The extendibility logic should capture the constraints we want to put on the relation $S$ in the abstract model to enforce extendibility behavior. As we saw in chapter 4, the extendibility relation is reflexive, and transitive, as it related a description or object to itself and all the descriptions that are more informative, with in the limit the whole object itself. Reflexivity and transitivity are well known constraints in modal logic, and the minimal modal logic of S4 describes such a system. In the isolated case (where we have no adjacency and aggregation) the only intriguing constraint we put on $S$ is that it has a top element when we look at each element in the relation. This top element is the whole object that, in the limit, a proper partial description (and the object itself) can be extended to. i.e.

- A partial description $a$ is extendable to all partial descriptions that partially describe the whole object that $a$ partially describes, and $a$ is also extendable to the most informative description of this object, the object itself.

This means that each element in the $S$ relation relates to itself and to all objects in the $S$-chain above, and that each element relates eventually to a unique local top element; i.e. then:

\textsuperscript{7}Note In fact we could have made up the concrete model with the adjacency alone. But then we would have lost that intuition.
6.2.8. **DEFINITION.** (Constraints for the extendibility logic)

- **(reflexivity)** \( \forall a \ aSa \)
- **(transitivity)** \( \forall a, b, c (aSb \land bSc \rightarrow aSc) \)
- **(top)** \( \forall a \exists b (aSb \land \forall c (aSc \rightarrow cSb)) \)
- **(unique)** \( \forall a, b, c ((aSb \land aS^c) \rightarrow \exists d (bSd \land cSd)) \)

The language for the extendibility logic becomes:

6.2.9. **DEFINITION.** (Language for the extendibility logic)

\[
L_{II} := \text{Prop} \cup L_{II} \sqcap L_{II} \sqcup L_{II} \sqcup L_{II} \sqcup L_{II}
\]

We will abbreviate \( \Delta := \neg \circ \neg \) to have convenient notation for the dual of \( \circ \). The interpretation is:

6.2.10. **DEFINITION.** (Semantics for the extendibility logic)

\[
M, s \models \circ A \iff \exists t (sSt \& M, t \models A)
\]
\[
M, s \models \circ^\cup A \iff \exists t (tSs \& M, t \models A)
\]

The 'top' constraint we have formulated for \( S \) is exactly expressed by the so-called McKinsey axiom, while the 'unique' constraint is the constraint that enforces the well known Church-Rosser property. The accompanying logic for both constraints (together with the reflexivity and transitivity) are respectively the \( S4.1 \) and the \( S4.2 \) modal logics ([HughesCresswell68]). The logic for extendibility will therefore be an \( S4.1+2 \) system.
6.2.11. Definition. (Axiomatics for the extendibility logic)

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

\[
\begin{align*}
(o\text{Distribution}) & \quad o(A \cup B) \rightarrow (oA \cup oB) \\
(o\cup\text{Distribution}) & \quad o^\cup(A \cup B) \rightarrow (o^\cup A \cup o^\cup B) \\
(o\text{Necessation}) & \quad \neg A \rightarrow \neg oA \\
(o^\cup\text{Necessation}) & \quad \neg A \rightarrow \neg o^\cup A
\end{align*}
\]

- then we need to add the rules to relate the modalities to their inverse modalities:

\[
(o\text{Inverse})A \rightarrow \neg o \neg o^\cup A
\]

- finally we need to add the axioms that reflect constraints on the extendibility relation (reflexivity, transitivity and McKinsey axiom).

\[
\begin{align*}
(T) & \quad A \rightarrow oA \\
(4) & \quad o o A \rightarrow oA \\
(M) & \quad \Delta oA \rightarrow o \Delta A \\
(C) & \quad o \Delta A \rightarrow \Delta oA
\end{align*}
\]

It is common knowledge in the field of modal logic that when we introduce the axioms for reflexivity \((T)\) and transitivity \(\ (4)\) for the modality \(o\), then transitivity and reflexivity can be deduced for its inverse modality \(o^\cup\). Thus we do not need to repeat the axioms \(T\) and \(4\) for \(o^\cup\). The extendibility modality enables one to say important things about partial descriptions and objects; e.g.

- an entity \(x\) is a description of an existing object (or an object itself) iff \(x\) satisfies \(o T\)
- an entity \(x\) is a description of an \(A\) object (or an \(A\) object itself) iff \(x\) satisfies \(o A\)

6.2.4 Aggregate logic

We have seen in the presentation of the language for categorial graphs that it is a core language element to talk about taking things together. This is called *aggregation*. In the aggregate logic we will shape the domain such that one can talk about aggregates. In other words we will be able to say things about things taken together. The structures of things taken together can be a number of things, among which are sets, multisets, and lists.
6.2. Modal languages

In our abstract model, the $Q$ relations model the aggregation. We do not constrain the behavior of the $Q$ relations yet. Curiously, this is not necessary because most things we want to say with a $\ast$ operation about taking things together do correctly coincide with the minimal intuition we could have about such a $\ast$ operation interpreted by a relation $Q$. Things will get more complicated when we start to require behavior of the different structures that are the result of the aggregation: sets, multisets, and lists. Then we need constraints that force associativity, multiplicity and order when the language enables us to say things about these matters. In the analysis these constraints on the abstract model will be optional.

For the aggregation we have different relations to interpret the different aggregation operations. We have separate families of relations for aggregations of objects ($Q^E$) and aggregations of partial descriptions ($Q^A$); and then for each type of structure, set, multiset and list again separate relations within each family ($Q^E_{\text{set}}, Q^E_{\text{multisets}},$ and $Q^A_{\text{list}}$ for $Q^E$ and $Q^A_{\text{set}}, Q^A_{\text{multisets}},$ and $Q^A_{\text{list}}$ for $Q^A$).

6.2.12. Definition. (Optional constraints for the aggregate logic) Let $Q$ be an arbitrary aggregation relation, then we can formulate the following constraint on $Q$ forcing associativity:

$$(\text{associativity}) \forall x (\exists y (xQyc \land yQab) \leftrightarrow \exists y' (xQay' \land y'Qbc))$$

Furthermore, we have constraints that are specific for certain structures, the sets, multisets, or lists. We already saw these constraints in the concrete models in chapter 4.

$$(\text{idem-consumption/cloning}) \forall a aQaa$$
$$(\text{commutativity}) \forall abc(aQbc \rightarrow aQcb)$$

The $Q$-relations for sets ($Q^E_{\text{set}}$ and $Q^A_{\text{set}}$) need to satisfy (at least) both idem-consumption/cloning and commutativity. The $Q$-relations for multisets ($Q^E_{\text{multisets}}$ and $Q^A_{\text{multisets}}$) need to satisfy (at least) commutativity and the negation\(^8\) of idem-consumption/cloning. Finally the $Q$-relations for lists ($Q^E_{\text{list}}$ and $Q^A_{\text{list}}$) need to satisfy (at least) both the negations of idem-consumption/cloning and the negation\(^9\) of commutativity.

\(^8\)Note, however, that it is not always necessary to require the negation of idem-consumption/cloning. Not requiring idem-consumption/cloning is enough to worry about multiplicity. Only when one strictly needs multisets, the negation of idem-consumption/cloning should be included. Cf. trace theory where there is a concept of partial commutativity.

\(^9\)Note, however, that it is not always necessary to require the negation of commutativity. Not requiring commutativity is enough to worry about order. Only when one strictly needs lists, the negation of commutativity should be included.
Although there are a lot more constraints needed to force the behavior of sets (such that aggregation is set-union), or multisets (such that aggregation is multiset-union), or lists (such that aggregation is list concatenation), the constraints from above are powerful enough to analyze logical systems for object oriented information systems that have the ability to express things about arity (counting) and order; two abilities we consider basic for a language for information systems in our analysis.

Let us now turn to the language for the aggregation logic. We introduce (dyadic) modal operators to talk about the (ternary) aggregation relations. In general there are three ways to talk about one ternary relation with a dyadic modal operator. We list the three cases for the aggregation relations:

1. you want to express that an object is the aggregation of two objects of a certain kind

2. you want to express that an object is the left part of an aggregation

3. you want to express that an object is the right part of an aggregation

The three modalities that express just these things form a 'versatile triple' and are studied in [Venema91] and [Benthem2000a]. A view like this is very relevant in the context of a graphical modeling language like the language of categorial graphs. When we depict a type (category) to be the aggregation of two other types (categories) we can visually 'see' all three ways to express a property for an aggregation (see figure 6.6). So it is only fair that we give the ability to express these different ways in the logic that is the meta language in which we are able to express all the things from the graphical language.

The language for adjacency logic now will look as follows:

\[\text{Figure 6.6: A category is the aggregation of two other categories.}\]

\[\text{10 Compare this with the two ways to talk about a binary relation with a monadic modal operator: the modality that traverses the relation from left to right, and its inverse, that traverses a relation from right to left.}\]
6.2.13. Definition. (Language for the aggregate logic)

\[ L_{III} := \text{Prop}[L_{III} \cap L_{III} \sqcup L_{III} \downarrow \neg L_{III}] \]
\[ \text{struct} := \text{set}|\text{multiset}|\text{list} \]
\[ X := E|A \]

The interpretation for the three modalities for the product then are interpreted as follows:

6.2.14. Definition. (Semantics for the adjacency logic)

\[ M, s \models A \ast_1 \text{struct} X B \text{ if } \exists t, u(sQ\text{struct} X tu \& M, t \models A \& M, u \models B) \]
\[ M, s \models A \ast_2 \text{struct} X B \text{ if } \exists t, u(tQ\text{struct} X su \& M, t \models A \& M, u \models B) \]
\[ M, s \models A \ast_3 \text{struct} X B \text{ if } \exists t, u(tQ\text{struct} X us \& M, t \models A \& M, u \models B) \]

For a reader that is familiar with the Lambek calculus, it may be enlightening to remark that there is a tight connection between the operations here and the operations of the Lambek calculus ([Lambek58]). In the Lambek calculus we also have three dyadic operations: one ‘*’ is the normal aggregation (i.e. like \( \ast_1 \)), and two operations ‘\( ' \) and ‘\( / ' \) that are respectively ‘left and right searching’. The latter \( A/B \) expresses the fact that if an entity gets aggregated to the right side with an entity of some type \( B \) they together (as an aggregate) form an object of some type \( B \). It is even true that we can define the Lambek slashes with the operations of the versatile triple and vice versa. E.g.

\[ A/B := \neg(\neg A \ast_2 B) \]
(i.e. it is not the case that I am aggregated to the right with a \( B \) entity and together we form an entity that is not of type \( A \))

\[ A \ast_2 B := \neg(\neg A/B) \]
(i.e. it is not the case that I am an entity that when concatenated to the right with a \( B \) entity, we form an entity that is not of type \( A \))

For the axiomatics of the aggregate logic we need a minimal modal logic for all the dyadic operators, together with the principles that connects the triples of related modalities. Moreover, we need to find the rules for the constraints mentioned at the beginning of this section.

6.2.15. Definition. (Axiomatics for the aggregate logic)
• all axioms and rules of propositional logic for the logic with \( \land, \lor \) and \( \neg \)

• rules for the dyadic modalities in the minimal dyadic modal logic (for all \( \text{struct} \in \{\text{set}, \text{multiset}, \text{list}\} \), \( X \in \{E, A\} \), \( i \in \{1, 2, 3\} \))

\[
\begin{align*}
(*_{\text{struct} X} \text{LeftDistribution}) & : (A \cup B) \cdot_{\text{struct} X} C \iff A \cdot_{\text{struct} X} C \cup B \cdot_{\text{struct} X} C \\
(*_{\text{struct} X} \text{RightDistribution}) & : (A \cdot_{\text{struct} X} (B \cup C) \iff A \cdot_{\text{struct} X} B \cup A \cdot_{\text{struct} X} C \\
(*_{\text{struct} X} \text{LeftNormal}) & : \neg (\bot \cdot_{\text{struct} X} A) \\
(*_{\text{struct} X} \text{RightNormal}) & : \neg (A \cdot_{\text{struct} X} \bot) \\
(*_{\text{struct} X} \text{LeftNecessation}) & : \neg A \iff A \cdot_{\text{struct} X} B \\
(*_{\text{struct} X} \text{RightNecessation}) & : \neg B \iff A \cdot_{\text{struct} X} B
\end{align*}
\]

• then we need to add the rules to relate the modalities of each versatile triple. These are the axioms to ensure that the three modalities can be interpreted by one \( Q \) relation from the different views. i.e. for all \( \text{struct} \in \{\text{set}, \text{multiset}, \text{list}\} \), \( X \in \{E, A\} \):  

\[
\begin{align*}
(*_{\text{struct}_1} \text{coherence}) & : A \cap (B \cdot_{\text{struct}_2} C) \rightarrow (B \cap (A \cdot_{\text{struct}_1} C)) \cdot_{\text{struct}_1} C \\
(*_{\text{struct}_2} \text{coherence}) & : A \cap (B \cdot_{\text{struct}_3} C) \rightarrow B \cdot_{\text{struct}_1} (C \cap (A \cdot_{\text{struct}_2} B)) \\
(*_{\text{struct}_3} \text{coherence}) & : B \cap (A \cdot_{\text{struct}_1} C) \rightarrow (A \cap (B \cdot_{\text{struct}_3} C)) \cdot_{\text{struct}_2} C \\
(*_{\text{struct}_1} \text{coherence}) & : B \cap (A \cdot_{\text{struct}_2} C) \rightarrow A \cdot_{\text{struct}_3} (C \cap (A \cdot_{\text{struct}_1} B)) \\
(*_{\text{struct}_2} \text{coherence}) & : C \cap (A \cdot_{\text{struct}_3} B) \rightarrow (A \cap (B \cdot_{\text{struct}_1} C)) \cdot_{\text{struct}_3} C \\
(*_{\text{struct}_3} \text{coherence}) & : C \cap (A \cdot_{\text{struct}_1} B) \rightarrow A \cdot_{\text{struct}_2} (B \cap (A \cdot_{\text{struct}_3} C))
\end{align*}
\]

This first coherence axiom relating \( *_1 \) to \( *_2 \) says that:

– when an entity is of type \( A \) and also \( (\cap) \) is an aggregation \( (*_1) \) of a \( B \)-entity and a \( C \)-entity, then this entity is an aggregation \( (*_1) \) of two entities:

- one entity that is both a \( B \)-entity and \( (\cap) \) an entity that is the left-hand part of an aggregation \( (*_2) \) of \( A \)-entity that has a \( C \)-entity as the right-hand part
- and another entity that is a \( C \)-entity

The other axioms relating the modalities of the versatile triple can be explained similarly.

• finally we formulate the axioms that reflect the optional constraints on the aggregation relations.

\[
\begin{align*}
(*_{\text{struct}_1} \text{associativity}) & : A \cdot_{\text{struct}_1} (B \cdot_{\text{struct}_1} C) \iff (A \cdot_{\text{struct}_1} B) \cdot_{\text{struct}_1} C \\
(*_{\text{struct}_1} \text{cloning}) & : A \rightarrow A \cdot_{\text{struct}_1} A \\
(*_{\text{struct}_1} \text{idem-consumption}) & : A \cdot_{\text{struct}_1} A \rightarrow A \\
(*_{\text{struct}_1} \text{commutativity}) & : A \cdot_{\text{struct}_1} B \rightarrow B \cdot_{\text{struct}_1} A
\end{align*}
\]
These constraints clearly correspond to the constraints we listed above. For example the \((\star_1^{\text{struct}}\times\text{commutativity})\) axiom says that when an object is an aggregation \((\star_1)\) of an \(A\)-entity and a \(B\)-entity then this object is also an aggregation \((\star_1)\) of a \(B\)-entity and an \(A\)-entity.

The language of the aggregate logic enables us to state a lot of powerful things about entities in our models. For example:

- **atomicity**: \(x\) is atomic iff \(x\) satisfies \(\neg(T \star_1^{\text{struct}} X \land T)\)
  (actually the conjunction of this formula for all the different types of aggregation; i.e. \(\neg(T \star_1^{\text{set} E} X \land T) \lor \neg(T \star_1^{\text{multiset} E} X \land T) \lor \neg(T \star_1^{\text{list} E} X \land T) \lor \neg(T \star_1^{A} X \land T)\))
- **membership**: \(x\) is a member of a struct structure iff \(x\) satisfies \(T \star_2^{\text{struct}} X \land T \lor T \star_3^{\text{struct}} X \land T\)
- \(x\) is a member of an \(A\) - type struct structure iff \(x\) satisfies \(A \star_2^{\text{struct}} X \land T \lor A \star_3^{\text{struct}} X \land T\)
- \(x\) is a member of a structure with an \(A\) - member iff \(x\) satisfies \(T \star_2^{\text{struct}} X A \lor T \star_3^{\text{struct}} X A\)

When we have *exchange* for the structures then it suffices to say for the membership:

- \(x\) is a member of a (multi)set iff \(x\) satisfies \(T \star_2^{\text{(multi)set} X} X \land T\)
- \(x\) is a member of an \(A\) - (multi)set iff \(x\) satisfies \(A \star_2^{\text{(multi)set} X} X \land T\)
- \(x\) is a member of a structure with an \(A\) - member iff \(x\) satisfies \(T \star_2^{\text{(multi)set} X} A\)

The above examples show that we can express some interesting *local* properties of entities in our model. The logic is even strong enough to express 'global properties'.

- **existential**: \(T \star_2^{\text{struct} X} A\) holds when \(A\) holds somewhere
  (Actually the disjunction of this formula for all the different types of aggregation; i.e. \((T \star_2^{\text{set} E} A) \lor (T \star_2^{\text{multiset} E} A) \lor (T \star_2^{\text{list} E} A) \lor (T \star_2^{\text{set} A} A) \lor (T \star_2^{\text{multiset} A} A) \lor (T \star_2^{\text{list} A} A)\))
- **universal**: \(\neg(T \star_2^{\text{struct} X}) \neg A\) holds when \(A\) holds everywhere
  (Again actually the disjunction of all the different types of aggregation)

Note that the logic of aggregation is surprisingly expressive when we realize that once we have these global expressions we can express the powerful principle of *induction* ([Benthem2000a]). We have the possibility to talk about atomicity and do existential and universal statements. Now let us denote 'atomicity' by \(\text{At} \).
Chapter 6. Logic of object oriented information

(At := ¬(T $\star$ struct X T), 'Existential' by Ex (Ex := T $\star$ struct X) and 'Universal' by Un (Un := ¬Ex¬), then we can express induction as follows:

\[ \text{Un}(A \rightarrow A) \cap \text{Un}(A \star X A \rightarrow A) \rightarrow \text{Un}A \]

The universal and existential modalities are well known from modal logic and we get them here for free (i.e. without axiomatizing them, but expressing them using the aggregation modalities). An interesting exercise for instance would be to deduce the S5 axioms for the defined existential and universal modalities from the earlier axioms for aggregation.

6.2.5 The combined system

In this system we combine all the features of the above logics. This means that the language contains all the operators mentioned above, and that the abstract model will inhabit all the constraints that the relations had in their isolated analysis. In the combined system we need to add the constraints and accompanying principles that regulate the interaction between the different subsystems. This means to regulate interaction between aggregation and adjacency, extendibility and aggregate and extendibility and adjacency.

Between aggregation and adjacency there are two important constraints that we also saw in chapter 4: regularity and extentiality.

Regularity says that the result object of taking two objects together should be consistent with the adjacents of the objects taken together. In a stronger version this constraint says that whenever we have objects a and b that have partial descriptions involving respectively objects c and d, then the aggregation of a and b, say e should have partial descriptions that also involve objects c and d.

Extentiality says that whenever an object has a partial description that is an aggregation of two other partial descriptions, then this object also has the two partial descriptions separately as partial descriptions.

6.2.16. Definition. (Constraints on the combined system for aggregation and adjacency)

(weak regularity)  \( aR_1 f \land fR_2 c \land bR_1 g \land gR_2 d \land eQ^{\text{struct}}Eab \rightarrow \exists i, (eR_1 h \land hR_2 c \land eR_1 i \land iR_2 d) \)

(strong regularity) \( aR_1 f \land fR_2 c \land bR_1 g \land gR_2 d \land eQ^{\text{struct}}Eab \rightarrow \exists i, j(Q^{\text{struct}}Ahi \land eR_1 j \land hR_2 c \land iR_2 d) \)

(extentiality) \( eR_1 j \land jQ^{\text{struct}}Ahi \rightarrow eR_1 h \land eR_1 i \)

$\Delta$
The weak regularity just states the existence of the appropriate partial descriptions when objects are taken together. The strong regularity also states that these partial descriptions should be aggregated, and by that relating the product of objects to the product of partial descriptions via the adjacency. This strong version is needed when we deal with resource conscious structures like multi-sets and sets, because then we need to ensure that the multiplicity is handled appropriately by the partial descriptions as well. Moreover we observe that the constraints 'strong regularity' and 'existentiality' clearly imply 'weak regularity', making it a redundant constraint (but it most directly translates to the textual-intuitively stated- regularity constraint).

Between the aggregation (product) of partial descriptions and the extendibility there is also some interaction. If an aggregation of two partial descriptions extends to some object, then it is a more informative description of the object than the partial descriptions it is an aggregation of. In general extendibility relates proper\(^{11}\) descriptions to more (or equal) informative proper descriptions or the object itself. This means that if a partial description \(a\) is an aggregation (product) of two partial descriptions \(b\) and \(c\), and \(a\) extends to an entity (object or partial description) \(d\), then also the partial descriptions \(b\) and \(c\) extend to \(d\)

6.2.17. DEFINITION. (Constraints on the combined system for aggregation and extendability)

\[(\text{more informative extendibility}) \quad a Q_{\text{struct}}^{A} b c \& a S d \rightarrow b S d \& c c S d\]

\(^{11}\)Proper means that the description is really a description of an object, i.e. it is a substructure of the adjacency structure of the object or it is the object itself; and thus not some aggregation of partial descriptions that is overstating the properties of the object it intends to describe.
Between Extendibility and adjacency there is the well expected constraint that says that if \( b \) is a partial description of \( a \) via the adjacency relation \( R_1 \), then \( b \) is extendable to \( a \).

**6.2.18. Definition. (Constraints on the combined system for adjacency and extendibility)**

\[ aR_1 b \rightarrow bSa \]

The language of the combined system includes the full set of modalities as we defined them in the beginning of this section (see definition 6.2.1). And also their interpretation is the one we started off with (see definition 6.2.2). The interesting question now is what are the axioms and rules relating to the combination of the various features.

**6.2.19. Definition. (Axiomatics of the combined system)**

- axioms and rules of the adjacency logic
- axioms and rules of the extendibility logic
- axioms and rules of the aggregate logic
- axioms and rules for combining adjacency and aggregation:
  - (weak regularity) \[ A \cap \lozenge_1 (F \cap \lozenge_2 C)^{struct_E} \rightarrow \lozenge_1 \lozenge_2 C \cap \lozenge_1 \lozenge_2 D \]
  - (strong regularity) \[ A \cap \lozenge_1 (F \cap \lozenge_2 C)^{struct_E} \rightarrow \lozenge_1 \lozenge_2 C \cap \lozenge_1 \lozenge_2 D \]
  - (\( \lozenge_1 \)extantiality) \[ \lozenge_1 (F)^{struct_A} \cap \lozenge_1 G \rightarrow \lozenge_1 F \cap \lozenge_1 G \]
6.2. Modal languages

- axioms and rules for combining aggregation and extendibility:

\[(\star^2_{\text{extendibility}}) \quad \text{T} \star^2_{\text{extendibility}} \circ A \rightarrow \circ A\]
\[(\star^3_{\text{extendibility}}) \quad \text{T} \star^3_{\text{extendibility}} \circ A \rightarrow \circ A\]

To explain the correspondence between the axioms and the constraints, look again at figure 6.7 illustrating the constraints between aggregation and adjacency. For convenience, we formulated the axiom such that we can read the picture in such that object a in the picture is of type A, and object b in the picture of type B etc.. The 'weak regularity' axiom now says: ”if I have an object that is the aggregation \((\star^f)\) of two objects

- one object that is of type A and \((\cap)\) that is an object that has a partial description \((\diamond_1)\) that is both of type \(F\) and \((\cap)\) is an adjacency witness \((\diamond_2)\) for an object of type C
- another object that is of type B and \((\cap)\) that is an object that has a partial description \((\diamond_1)\) that is both of type G and \((\cap)\) is an adjacency witness \((\diamond_2)\) for an object of type D

then this object has a C-object as adjacent \((\diamond_1\diamond_2)\) and \((\cap)\) has a D-object as adjacent \((\diamond_1\diamond_2)\).” The other correspondences can be explained similarly.

- axioms and rules for combining extendibility and adjacency:

\[(\diamond_1 \circ \text{overlap}) \quad \diamond^U_1 A \rightarrow \circ A\]

\(\Delta\)

From this system we can deduce non trivial principles for the categorial graphs. For example "partial descriptions of an A object extend to an A object":

\[
\frac{\diamond^U_1 A \rightarrow A (\diamond_1 \circ \text{Overlap})}{\Box_1 (\diamond^U_1 A \rightarrow A)} \quad (\Box_1 \text{Necessation})
\]
\[
\frac{A \rightarrow \Box_1 \diamond^U_1 A (\diamond_1 \text{Inverse})}{\Box_1 \diamond^U_1 A \rightarrow \Box_1 \circ A} \quad (\Box_1 \text{Distribution})
\]
\[
\frac{A \rightarrow \Box_1 \circ A}{(\text{cut})}
\]

The combined system gives us the full power of the categorial graph meta language of chapter 4, and even a little more, because we introduced additional operators that take a look at things from another perspective. These are:

- We added extendibility also in the language
- split the adjacency in two steps
- added inverse modalities for the unary modalities of adjacency and extendibility
Chapter 6. Logic of object oriented information

<table>
<thead>
<tr>
<th>description</th>
<th>meta language</th>
<th>modal language</th>
</tr>
</thead>
<tbody>
<tr>
<td>conjunction</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>disjunction</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>negation</td>
<td>¬</td>
<td>¬</td>
</tr>
<tr>
<td>adjacency</td>
<td>◊</td>
<td>◊₁ ◊₂</td>
</tr>
<tr>
<td>aggregation</td>
<td>*</td>
<td>split into collections of operations</td>
</tr>
<tr>
<td>reflection</td>
<td>self</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure 6.9: Correspondence between the categorial graph meta language and the combined modal language

- we introduced the modalities for the dyadic aggregation that look at the aggregation from a different perspective (versatile triple).

Correspondence between the categorial graph meta language of chapter 4 and the pure modal language are summarized in the figure 6.9.

6.2.6 Conclusion

We are now in the business of modal logics, and can import all kinds of techniques: decidability, axiomatization, frame correspondence, bisimulation. We say that the natural constraints on the abstract model correspond nicely with modal formulas. This enables one to study frames corresponding to interesting principles like □₁□₁⊥, or □₁T ↔ ◊₂⊥. This way we give something back to the modal logic community: the motivation to study certain principles based on the very practical case of object modeling.

6.3 Other logical formalizations

Starting from the modal logic developed in this chapter we can traverse to other logical formulations. The modal logic community has seen several nice correspondences with more general and more specialized logics. In this section we will explore such paths.

6.3.1 Translation

Modal languages are part of first order logic (speaking generally; we could however add some higher operations involving fixed-points). The language constructs of modal logic can be translated into general first order logic using a standard translation. This translation will show that all the principles we defined for the modal logic for categorial graphs are first order definable. The modal language of categorial graphs can be translated to the following first order language:
6.3. Other logical formalizations

6.3.1. Definition. (First order language for categorial graphs) Recall the definition of the modal language in definition 6.2.1. The first order language for categorial graphs $L^{FO}$ contains the following:

- For all propositional variables in the modal language $L$, $P \in \text{Prop}$, we have a unary predicate $P^{FO}$ in $L^{FO}$

- For all modalities $m$ of arity $i$ we define predicates $T_m$ of arity $i + 1$ in $L^{FO}$, i.e.
  
  $T_{\emptyset_1}$ of arity 2 for $\emptyset_1$
  $T_{\emptyset_2}$ of arity 2 for $\emptyset_2$
  $T_{\emptyset_1'}$ of arity 2 for $\emptyset_1'$
  $T_{\emptyset_2'}$ of arity 2 for $\emptyset_2'$
  $T_\circ$ of arity 2 for $\circ$
  $T_{\circ'}$ of arity 2 for $\circ'$
  $T_{\circ^\text{struct}}$ of arity 3 for $\circ^\text{struct}$

  (struct $\in \{\text{set, multiset, list}\}$, $i \in \{1, 2, 3\}$)

- all other standard things to complete first order language including variables, $\land$, $\lor$, and $\neg$

$\Delta$

The (standard) translation for this language now looks as follows:

6.3.2. Definition. (standard translation) Let $x, y, y_1, y_2$ be individual variables in the first order language $L^{FO}$. The standard translation $ST_x$ taking formulas from $L$ into $L^{FO}$ is defined as follows:

$$
ST_x(P) = Px
$$
$$
ST_x(\bot) = x \neq x
$$
$$
ST_x(\top) = x = x
$$
$$
ST_x(\lnot A) = \lnot ST_x(A)
$$
$$
ST_x(A \land B) = ST_x(A) \land ST_x(B)
$$
$$
ST_x(A \lor B) = ST_x(A) \lor ST_x(B)
$$
$$
ST_x(\emptyset_1(A)) = \exists y(T_{\emptyset_1}xy \land ST_y(A))
$$
$$
ST_x(\emptyset_2(A)) = \exists y(T_{\emptyset_2}xy \land ST_y(A))
$$
$$
ST_x(\emptyset_1'(A)) = \exists y(T_{\emptyset_1'}xy \land ST_y(A))
$$
$$
ST_x(\emptyset_2'(A)) = \exists y(T_{\emptyset_2'}xy \land ST_y(A))
$$
$$
ST_x(\circ(A)) = \exists y(T_{\circ}xy \land ST_y(A))
$$
$$
ST_x(\circ'(A)) = \exists y(T_{\circ'}xy \land ST_y(A))
$$
$$
ST_x(\circ^\text{struct}(A)) = \exists y(T_{\circ^\text{struct}}xy \land ST_y(A))
$$
$$
ST_x(A \circ^\text{struct} B) = \exists y_1, y_2(T_{\circ^\text{struct}}xy_1y_2 \land ST_{y_1}(A) \land ST_{y_2}(B))
$$

(\text{where } y_1, y_2 \text{ are fresh variables})

$\Delta$
As an example look at the following modal formulas and their translations

\[ ST_x(\Diamond_1 p \rightarrow \Diamond_2 p) = (\exists y (T_0_1 xy \land P_y)) \rightarrow (\exists y (T_0_1 xy \land P_y)) \]
\[ ST_x(\Box_1 \Box_1 \perp) = \forall y_1 (T_0_1 xy_1 \rightarrow \forall y_2 (T_0_1 y_1 y_2 \rightarrow y_2 \neq y_2)) \]

We can optimize the translation from above by translating related modalities like \( \Diamond_1 \) and \( \Diamond_2 \) or a versatile triple \( \ast_1^{setE}, \ast_2^{setE}, \ast_3^{setE} \) to the same predicate with different order in the variables. i.e.

\[ ST_x(\Diamond_1 (A)) = \exists y (T_0_1 xy \land ST_y(A)) \]
\[ ST_x(\Diamond_2 (A)) = \exists y (T_0_1 yx \land ST_y(A)) \]
\[ ST_x(A \ast_1^{structE} B) = \exists y_1, y_2 (T_0_1 xy_1 y_2 \land ST_y(A) \land ST_y(B)) \]
\[ ST_x(A \ast_2^{structE} B) = \exists y_1, y_2 (T_0_1 xy_1 y_2 \land ST_y(A) \land ST_y(B)) \]
\[ ST_x(A \ast_3^{structE} B) = \exists y_1, y_2 (T_0_1 xy_1 y_2 \land ST_y(A) \land ST_y(B)) \]

### 6.3.2 First Order approach

In this setting we can go *up* in terms of generality to the first order level. This means that we can do simple first order logic by translating our modal formula's into first order formulas using the translation from above, and then interpret these first order formulas in standard Tarski models for first order logic. If we translate all the principles from the modal logic into the first order logic, we have a proper first order logic for categorical graphs. In order to make the Tarski models more concrete we can introduce sorts, a sort for objects, and a sort for partial descriptions. Then we can study the first order properties in a well known type of models: two sorted Tarski models. We will use this fact to say something about the completeness and complexity in future sections.

### 6.3.3 Resource approach

Probably more relevant is going *down* in terms of generality. We will gain in specificness by considering calculi in which we can talk directly about resources. For this we will use sequent calculi and leave out (or diversify on) the structural rules. In other words we will use a substructural calculus. When, in the abstract case, we leave in all the operations we actually gain in expressiveness, and therefore in complexity as well. We will look, instead, at tractable fragments of modal logics found in categorial or linear logic using the extra expressiveness. We will specifically look at a minimal language that has product and conjunction and adjacency.

#### 6.3.3. Definition. (Substructural Adjacency language) Let \( L_{SA} \) be the language called *substructural adjacency language* that talks about structures and adjacency; i.e. about aggregating (taking things together) objects and partial descriptions in a resource conscious manner.

\[ L_{SA} = Prop \setminus L_{SA} \ast L_{SA} \setminus L_{SA} \setminus \Diamond_1 L_{SA} \setminus \Diamond_2 L_{SA} \]
The calculus is a simple substructural one. The language of structures consists of formula’s. In a sequence calculus, like the one below, we deal with terms, which are comma separated lists (possibly empty) of formulas.

6.3.4. DEFINITION. (Calculus for the substructural adjacency logic)

\[
\begin{align*}
(AX) & \quad A \Rightarrow A \\
(CUT) & \quad \Gamma \Rightarrow A \quad \Gamma', A \Rightarrow B \\
& \quad \Gamma, \Gamma' \Rightarrow B \\
(L\Gamma) & \quad \Gamma, A \Rightarrow C \\
& \quad \Gamma, A \cap B \Rightarrow C \\
& \quad \Gamma, A \cap B \Rightarrow C \\
(R\Gamma) & \quad \Gamma \Rightarrow A \quad \Gamma \Rightarrow B \\
& \quad \Gamma \Rightarrow A \cap B \\
(L\ast) & \quad \Gamma, A, B \Rightarrow C \\
& \quad \Gamma, A \ast B \Rightarrow C \\
(R\ast) & \quad \Gamma \Rightarrow A \quad \Gamma' \Rightarrow B \\
& \quad \Gamma, \Gamma' \Rightarrow A \ast B
\end{align*}
\]

For the modalities we add the following:

\[
\begin{align*}
(\diamond I_1) & \quad A \Rightarrow B \\
& \quad \diamond_1 A \Rightarrow \diamond_1 B \\
(\diamond I_2) & \quad A \Rightarrow B \\
& \quad \diamond_2 A \Rightarrow \diamond_2 B
\end{align*}
\]

We could now say things resource consciously:

- In Model $\mathcal{M}$ an $A$ object has (at least) two partial descriptions, a $B$ and a $C$ one
  $\mathcal{M} \models A \Rightarrow \diamond_1 B \ast \diamond_1 C$

- In Model $\mathcal{M}$ an $A$ object has (at least) two adjacents, a $B$ and a $C$ one
  $\mathcal{M} \models A \Rightarrow \diamond_1 \diamond_2 B \ast \diamond_1 \diamond_2 C$

More specifically about structures one can say:

- in model $\mathcal{M}$ the structures are order unconscious for $A$ items
  $\mathcal{M} \models A \ast T \Rightarrow T \ast A$

- in model $\mathcal{M}$ the structures are order unconscious for $A$ and $B$ items
  $\mathcal{M} \models A \ast B \Rightarrow B \ast A$

- in model $\mathcal{M}$ the structures could sometimes be counting conscious for $A$ items
  $\mathcal{M} \not\models A \Rightarrow A \ast A
We could take it a step further and introduce more reasoning connectives plus the remaining modalities: negation, disjunction and extendibility. To be complete (and very similar to the calculus of the concrete model of chapter 4) we list the remaining language items and rules.

6.3.5. DEFINITION. Let $L_{FCA}$ be the language called full substructural categorial language

$$L_{FCA} = L_{SA} \cup L_{FCA} \cup L_{\neg FCA} \cup L_{FCA}$$

6.3.6. DEFINITION. (Calculus for the full substructural adjacent language)

The rules of the substructural adjacency language plus the following:

$$
\begin{align*}
(L\cup) & \quad \frac{\Gamma, A \Rightarrow C \quad \Gamma, B \Rightarrow C}{\Gamma, A \cup B \Rightarrow C} \\
(R\cup) & \quad \frac{\Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow A \cup B} \\
(L\neg) & \quad \frac{\Gamma \Rightarrow A, \Delta}{\Gamma, \neg A \Rightarrow \Delta} \\
(R\neg) & \quad \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \neg A, \Delta}
\end{align*}
$$

For the modalities we add the following:

$$
\begin{align*}
(oI) & \quad \frac{A \Rightarrow B}{\circ A \Rightarrow \circ B} \\
(\circ_1 Distribution) & \quad \circ_1 (A \cup B) \Rightarrow \circ_1 A \cup \circ_1 B \\
(\circ_2 Distribution) & \quad \circ_2 (A \cup B) \Rightarrow \circ_2 A \cup \circ_2 B \\
(oDistribution) & \quad \circ (A \cup B) \Rightarrow \circ A \cup \circ B
\end{align*}
$$

The logics in these examples capture quite a bit of the expressive power we want to have for talking about objects. Such fragments therefore are valuable for doing specific reasoning in a computationally more tractable setting than the more general approaches. We will talk about reasoning and complexity matters in a moment.

6.4 Axioms and completeness

Here we will investigate the completeness of the axiomatics with respect to the abstract models and the intended models that are presented above. Why would we bother to investigate completeness? In other words what does completeness mean for our languages and our models?
If we have a semantically specified logic, then completeness w.r.t. some calculus means that we have found a calculus that syntactically (and exactly) characterizes this logic. Also if we have a syntactically specified, then logic completeness w.r.t. a semantics means that we have found a semantic characterization of this logic.

6.4.1 The first-order case

For the first order logic for categorial graphs (the logic we get after translation) we can derive some results from the general framework of first order logic when it comes to matters of axiomatization and completeness.

For the abstract models we know that all the logics we presented above are effectively axiomatizable, because the constraints we put on the relations of the abstract models are all first order definable (we gave these definitions above). Completeness then is trivial. On the other side we also know that the logic system will most likely be undecidable, unless we are really so lucky that the set of constraints mitigated the undecidability of first order logic in the general (unconstrained) case. We do not believe that this is the case, and are of the opinion that a lot more can be gained in this matter when we look *down* to less general frameworks.

For the intended models we have bad news in the first order case right away. The logics of categorial graphs will be at least as complex as 'True Arithmetic' when we have the aggregations in our model as a structure domain (product structure). This follows directly from a result of Quine ([Quine46], [Benthem91]) that states that the first order theory of simple syntax\(^{12}\) is equivalent to True Arithmetic. To be precise quote the result:

**6.4.1. Theorem. (Quine) Consider the model \(M\{a, b\} \) containing a binary operation of concatenation\(^{13}\) and all finite strings from the two-symbol alphabet \(\{a, b\}\). The first order theory of \(M\{a, b\}\) is equivalent to 'True Arithmetic' (\(\Omega(\mathbb{N}, +, \cdot)\)).**

True Arithmetic is known to be undecidable, non-axiomatizable and of very high complexity by Gödel's and Tarski's classical theorems. The reason for such absurdly high complexity is that we have already too much structure when we have a structure domain (or simple syntax for that matter) when we allow the full first order language to talk about it. This structure can be 'abused' (with the expressive power of first order language) to code numbers and do arithmetic in it. We say 'abused', because the structures were not meant to code arithmetic but only complex information structures.

\(^{12}\)Syntax that can be concatenated to form syntax again; i.e. what we have in a structure domain.
\(^{13}\)or for that matter, a ternary relation of concatenation
6.4.2 The modal case

In the presentation of the different logics we were quite easy about the correspondence between the constraints and the accompanying axioms. This has a reason, because from the field of modal logic we know that such correspondences are generally valid when the modal axioms have a suited syntactical form; namely the Sahlqvist form ([BlackburnRijkeVenema01]).

6.4.2. DEFINITION. (Very Simple Sahlqvist Formula\textsuperscript{14})

- An occurrence of a proposition letter \( P \) is a positive occurrence if it is in the scope of an even number of negation signs. A modal formula is positive if all occurrences of its proposition letters are positive.

- A very simple Sahlqvist antecedent in a modal language is a formula built up from \( \top, \bot, \) and proposition letters, using only \( \sqcap \), and existential modalities (e.g. the \( \Diamond \) but not its dual \( \Box \)).

- A very simple Sahlqvist formula is an implication \( A \rightarrow B \) where \( A \) is a very simple Sahlqvist antecedent and \( B \) is a positive formula.

\[\Box\]

For example it is easy to see that that the \( (*_{1} \text{struct}X, *_{2} \text{struct}X \text{ coherence}) \) axiom is a very simple Sahlqvist formula. When we look at

\[ A \sqcap (B *_{1} \text{struct}X C) \rightarrow (B \sqcap (A *_{2} \text{struct}X C)) *_{1} \text{struct}X C \]

we see that the antecedent is built from proposition letters \( A, B, \) and \( C, \) and the connective \( \sqcap \) and the existential dyadic modality \( *_{1} \text{struct}X, \) hence it is a very simple Sahlqvist antecedent. Moreover all occurrences of \( A, B \) and \( C \) in the conclusion are positive.

6.4.3. THEOREM. (Sahlqvist correspondence theorem) Let \( A \) be a Sahlqvist formula for a modal language \( L \). Then \( A \) corresponds to a first order condition \( c_{A} \) on the models of \( L \). Moreover \( c_{A} \) is effectively computable from \( A \).

General modal logic provides an even stronger result. In the light of axiomatics and completeness these Sahlqvist formulas have the following property.

6.4.4. THEOREM. (Sahlqvist completeness theorem) Given a set of Sahlqvist axioms \( \Sigma \), the minimal normal modal logic \( K \) extended by the axioms of \( \Sigma \) is complete with respect to the models that satisfy all the corresponding first order conditions.

\textsuperscript{14} Actually a broader syntactically characterized collection of formulas, the Sahlqvist formulas, will have the same desirable properties as mentioned in the succeeding theorems as these very simple ones. The 'very simple' subset suffices for our purposes.
6.4. Axioms and completeness

For all but one of the axioms for the logics of categorial graphs from above we can easily establish that they are Sahlqvist formulas. Thus for the logics restricted to those axioms and corresponding conditions we have that

- the axioms correspond to the accompanying first order constraints
- the axiomatization is complete with respect to the abstract model

All the constraints that we encountered for the categorial graph logic correspond to Sahlqvist formulas, except one: the McKinsey axiom

\[(M) \rightarrow \neg \neg A \rightarrow \neg A\]

corresponding to the local top condition

\[(\text{top}) \forall a \exists b (a S b \& (\forall c : a S c \rightarrow c S b))\]

A normal modal logic with the McKinsey axiom added (i.e. the modal system S4.1), however, is a complete axiomatization with respect to a relational structure with the 'top' condition. For proving completeness for the combined systems now the question remains whether or not adding Sahlqvist formulas to the system with the McKinsey axiom (S4.1) frustrates the completeness (note that adding Sahlqvist formulas to complete normal logics that already contain Sahlqvist formulas does leave completeness intact). To our knowledge this question has not been answered yet, meaning that we cannot solve this concrete matter with standard results of modal logic. Therefore it remains an open question here, of which we strongly believe in the positive answer.

For the intended models the situation is more complex (and we can far less rely on the known results in modal logic). Similarly, as in the first order case, the danger of too much structure resulting in an inherently incomplete system is possible. The structure of the domain is as complex as in the first order case. The modal language, on the contrary, is normally less expressive as the full first order language, so there is still a chance to get completeness here. Nevertheless there are a lot of very powerful things that one can say in the modal language about the mathematical structure of the intended model (for a thorough investigation in these matters see [Benthem2000a]). An example that we have seen in the presentation of the aggregation logic is that we can express the induction principle. This indicates that the completeness issue could be a very tricky one, because we can express very complicated mathematical structures using induction.

So how are the chances for interesting completeness theorems here? This is not known yet and could be an interesting open question for logicians.
6.4.3 The resource case

The previous analysis points out that our models can give rise to completeness proofs that are more complex than in traditional modal logic. But it can be the other way around: completeness proofs for our models can also turn out to be a lot simpler than what we normally see in modal logic. We will demonstrate this by the results for the resource fragments of the logics for categorial graphs. Now we again take a step further *down* in generality and see what we can say about completeness for the substructural fragments we have seen above.

For the abstract models we can, in relation to the resource language and sequent calculus that is present in the resource case, prove completeness using the 'minimalistic' strategy of proving completeness for resource logics that is investigated by Kurtonina ([Kurtonina95]), Buzkowskii ([Buszkowski86]), and Dosen ([Dosen88], [Dosen89]). We will prove completeness for the calculus of the substructural adjacency language $L_{SA}$ with the adjacency modalities ($\bigtriangleup_1, \bigtriangleup_2$), conjunction ($\cap$) and aggregation ($\ast$).

6.4.5 Theorem. The calculus of $L_{SA}$ is complete with respect to the abstract models $\mathcal{M} = < U, Q, R_1, R_2, V >$

Proof: To prove completeness we construct a canonical model as follows:

1. Universe $U = \{ A | A \text{ is a formula of } L_{SA} \}$
2. Product relation $Q$ is defined as
   \[ AQCD \iff \forall E, F(C \Rightarrow E \land D \Rightarrow F \text{ then } A \Rightarrow E \ast F) \]
3. Adjacency relations $R_1$ and $R_2$ are defined as
   \[ AR_1C \iff \forall D(\vdash C \Rightarrow D \text{ then } \vdash A \Rightarrow \bigtriangleup_1 D) \]
   \[ AR_2C \iff \forall D(\vdash C \Rightarrow D \text{ then } \vdash A \Rightarrow \bigtriangleup_2 D) \]
4. Valuation $V$ is defined as $V(P)\{A|A \rightarrow P\}$ for propositions $P \in L_{SA}$

For the canonical model $\mathcal{M}$ we prove the truth lemma:

$\mathcal{M}, A \models B$ iff $\vdash A \Rightarrow B$

- **CASE $B = P$** for a proposition $P$: directly from the definition of valuation $V$

- **CASE $B = C \cap D$**:
  \[ \vdash A \Rightarrow C \cap D \] iff \[ \vdash A \Rightarrow C \text{ and } \vdash A \Rightarrow D \]
  iff (by induction) $\mathcal{M}, A \models C$ and $\mathcal{M}, A \models D$
  iff $\mathcal{M}, A \models C \cap D$
6.4. Axioms and completeness

- CASE $B = \Theta_1 C$:

  \[ \vdash A \Rightarrow \Theta_1 C \quad \text{then} \quad AR_1 C \text{ because } \forall D \text{ if } \vdash C \Rightarrow D \text{ then} \]
  \[\begin{array}{c}
  C \Rightarrow D \\
  A \Rightarrow \Theta_1 C \\
  \Theta_1 C \Rightarrow \Theta_1 D \\
  \hline
  \vdash \Theta_1 D \\
\end{array}\]

  then $\mathcal{M}, A \models \Theta_1 C$ (by definition semantics)

- CASE $B = \Theta_2 C$:

- CASE $B = C \ast D$:

  \[ \vdash A \Rightarrow C \ast D \quad \text{then} \quad AQBC \text{ because } \forall E, F \text{ if } \vdash C \Rightarrow E, \vdash D \Rightarrow F \]
  \[\begin{array}{c}
  C \Rightarrow E \\
  D \Rightarrow F \\
  A \Rightarrow C \ast D \\
  C \ast D \Rightarrow E \ast F \\
  \hline
  \vdash E \ast F \\
\end{array}\]

  then $\mathcal{M}, A \models C \ast D$ (by definition semantics)

Hence we have completeness.

If we add more connectives (i.e. disjunction and negation) this simple construction will not work as it stands, and the canonical model will become more complex. We could then, for example, use the The Dosen strategy ([Dosen89]) which introduces an additional relation $\leq$ over the aggregation structure to handle the aggregation in relation to the disjunction\footnote{If we aggregate something of type $A \cup B$ with something else, we cannot just look at the cases where we either have an $A$ object or a $B$ object in the aggregation.}. This relation is a technical one, which we can not really give a proper meaning, because in the abstract model it enables us to characterize objects that are of indefinite type\footnote{i.e. we need an object $x$ that is of type $A \cup B$ in order to canonically interpret $(A \cup B) \ast C$, but in the canonical model $x$ is neither of type $A$ or of type $B$ (it is indefinite).}. We could extend our ternary relation $Q$ that models the aggregation such that it covers this $\leq$ to handle indefinite objects as well. This is realized by letting $Q$ model $a \cdot b \leq c$ instead of $a \cdot b = c$ as before. Note however that with this technicality we drift further away from our intended model, because there we do not have these indefinite objects.
Let us now look at the intended models. In the literature there is only one result, to our knowledge, that proves completeness in a resource logic (i.e. products) w.r.t. a concrete model. This is the famous completeness proof of Pentus ([Pentus93]) for the (multiplicative) Lambek calculus. Completeness for a concrete model of strings for the rules of the Lambek calculus follows from the following:

**6.4.6. Theorem.** (Pentus) The recognizing power of the (multiplicative) Lambek calculus is precisely the class of all context free languages.

Pentus proves this by showing that for every given Lambek grammar (i.e. a collection of formulas recognizing all the formulas we can infer from them, seen as strings) we can effectively construct a coinciding categorial grammar and vice versa. This means that the Lambek calculus is complete for a model of strings. This proof is very involved, even though the Lambek calculus itself is quite limited. And thus it seems, at this time, extremely hard to establish a similar result for the concrete and more involved models for categorial graphs. Pentus's result however, implies that the basic logic of aggregation is complete for intended sequence models, so a completeness result for a richer language is a serious possibility. We hope that the challenge to find it will be taken up by the logic community.

We have seen a number of calculi for our languages for categorial graphs, and nice completeness results for these calculi with respect to the abstract models. These abstract models properly show the behavior we are interested in when talking about information objects. In this respect we have 'good' calculi for reasoning about these interesting matters. There is, however, still a gap between the abstract models and the intended models, and this gap is not bridged by a completeness proof of a calculus with respect to the intended models. Judging by the literature the completeness issue for intended (rich) structures is a very hard one and has only a few positive results (we already mentioned the completeness proof of Pentus for the Lambek calculus with respect to languages). The completeness issue for object models raised here is yet another interesting challenge to logicians who like to think about a real structure of current interest.

---

17Even better would be a Sahlqvist-like result for resource logics. Then we would have a similarly strong tool for solving completeness and definability questions as for normal modal logic. We have seen above that the Sahlqvist theorems for normal modal logic were very helpful for logical engineering of object oriented intuitions.
6.5 Complexity

In this part we analyze the complexity of the logics for categorial graphs. The complexity gives a measure for the 'hardness' of computational tasks using the expressive power of the systems presented here for object orientation.

6.5.1 Benchmark tasks

There are several 'benchmark' tasks for which computational analysis is done on logics. These tasks are the basis of most of the important algorithmic solutions for computing with the logics analyzed, and thus by assuming generality of these logics, they are indicative for many nontrivial computational tasks of the application domain the logic talks about. These tasks are:

- **Model checking.** Given a model $M$, and an entity $x \in M$, and a formula $A$ in the logical language $L$, then the task of model checking consists of checking whether $M, x \models A$. This task is measured in the amount of computational steps in terms of the size of $M$ plus the size of $A$.

- **Satisfiability (SAT).** Given a formula $A$ in $L$, does there exist a model $M$ and an entity $x \in M$ such that $M, x \models A$? The complexity of this task is measured in terms of the size of $A$.

- **Inference.** Given formulas $A, B$ in $L$, can we proof $A \vdash B$ from the calculus of $L$? This task is measured terms of the size of $A$ plus the size of $B$.

We will analyze these tasks for the logics of categorial graphs below. In our case the complexity of these benchmark tasks are indicative for many of the non-trivial computational tasks in object oriented models.

6.5.2 Model checking

Model checking is a common task in working with information systems. It is the question whether an expression (say constraint) is satisfied in some part of the information system.

The complexity of this task is easily proven to be in $P$ (class of polynomial time computable tasks) for all the abstract modal logics of categorial graphs we presented here. For model checking we need to verify that a formula $A$ is satisfied in a given entity $x$ of our model $M$. This involves (worst case)

1. when $A$ is a proposition letter a check of the valuation in $x$ which is bound by the size of $M$

2. when $A$ is $B \cap C$ the check for $M, x \models B$ followed by the check for $M, x \models C$ which by induction on the structure of the formula are polynomial time computable in the size of $M$ plus the size of $A$
3. when $A$ is $\neg B$ the check for $\mathcal{M}, x \models B$ followed by the inverse conclusion, which by induction on the formula structure is polynomial time computable in the size of $\mathcal{M}$ plus the size of $A$

4. when $A$ is $\Diamond B$ (where $\Diamond$ is any of the monadic modalities) then $\forall y(xRy)$ (where $R$ is the relation interpreting the modality $\Diamond$) we need to check $\mathcal{M}, y \models B$. The individual checks are, again by induction on the structure of the formula, polynomial time computable. The number of checks (i.e. number of $y$'s) is bound by the size of $\mathcal{M}$ (number of accessible worlds). Hence we are again polynomial time computable in the size of $\mathcal{M}$ plus the size of $A$.

5. when $A$ is $B \ast C$ (where $\ast$ is any of the dyadic modalities) then $\forall y, z(xQyz)$ (where $Q$ is the relation interpreting $\ast$) we need to check $\mathcal{M}, y \models B$ and $\mathcal{M}, y \models C$. The individual checks are by induction on the structure of the formula, polynomial time computable. The number of checks is bound by the square of the size of $\mathcal{M}$ (all possible pairs of accessible worlds of $\mathcal{M}$). Hence we have again polynomial time computability in the size of $\mathcal{M}$ plus the size of $A$ for model checking.

For the intended models and the resource models we need to take some extra care in analyzing the complexity of model checking. These models are of infinite size, because they contain free product structures (the domain structures). For example we have objects $x \cdot \ldots \cdot x$ of arbitrary length in our models. This complicates the complexity analysis as we defined it (and as it is commonly defined) for model checking, because we measure complexity in the size of the model. The solution to analyze these infinite systems is to measure their complexity in the size of a finite generator of the models. The infinite product structures (for aggregation) are generated from a finite set of atomic entities, and also the condition of regularity enforces that this base of atomic entities really completely generates the model. We discussed the generator of the intended models already in chapter 4. The generator consists of the atomic objects of the model together with the relations that specify the adjacency structure. Summarizing, we will measure the complexity of model checking in the size of the formula $|A|$ plus the size of the generator $|\text{gen}(\mathcal{M})|$ of the model $\mathcal{M}$ and the size of the entity $x$ of $\mathcal{M}$ in which we do the model checking (we need to add $x$ here because unlike in the former case $x$ (in general) is not part of $\text{gen}(\mathcal{M})$ and therefore can obfuscate the complexity analysis when it is taken very large).

The complexity for model checking for the intended models and the resource models is harder then P-time. We will prove that it is NP-hard. The reason for this increase is that an object can be split into parts in a number of ways that is exponential to the size of the object.
6.5.1. **Theorem.** Model checking for the resource logic for categorial graphs \((L_{FCA})\) is NP-hard

**Proof:** We will prove NP-hardness by a reduction from the ‘exact cover’ problem to model checking for the resource logic. The ‘exact cover’ problem is well known to be NP-complete ([GareyJohnson79]). The ‘exact cover’ problem is the following:

Given a set \(X\) with \(x\) elements and subsets \(A_1, \ldots, A_n\) of \(X\), is there a collection of \(k\) subsets \(A_{i_1}, \ldots, A_{i_k}\) that exactly covers \(X\) (i.e. the union of the \(A_i\)’s contain all elements of \(X\) precisely once)?

We can reformulate this problem in terms of model checking for the resource logic as follows:

1. We express that a set \(X\) has elements \(p_1, \ldots, p_x\) by the formula \(P_1 *_1 P_2 * \cdots * P_x\). Let us abbreviate this formula by \(\overline{X}\). To make this scheme work we need to require that in our model all \(P_i\) are satisfied in different entities\(^{18}\)

2. Similarly we can express that a subset \(A_i\) contains certain elements; i.e. \(\overline{A_i} := P_{i_1} * \cdots * P_{i_k}\).

3. Now we can express the set \(X\) is exactly covered by the \(k\) subsets \(A_i\) by stating

\[
\overline{X} \cap [(\overline{A_{i_1}} \cup \cdots \cup \overline{A_{i_k}}) * \cdots * (\overline{A_{i_1}} \cup \cdots \cup \overline{A_{i_n}})]
\]

\( \cdots k \) times \( \cdots \)

Let us now choose a model \(\mathcal{M}\) with an element \(X\) that is the aggregate of all its \(x\) members of atomic objects, and let us choose \(n\) subsets characterized by \(A_1, \ldots, A_n\). Now it is the case that \(X\) has an exact cover of \(k\) subsets \(A_{i_1}, \ldots, A_{i_k}\) if and only if

\[
\mathcal{M}, X \models \overline{X} \cap [(\overline{A_{i_1}} \cup \cdots \cup \overline{A_{i_k}}) * \cdots * (\overline{A_{i_1}} \cup \cdots \cup \overline{A_{i_n}})]
\]

\( \cdots k \) times \( \cdots \)

\(\square\)

6.5.3 **Satisfiability**

In the context of information systems using the theory of categorial graphs, satisfiability answers questions for the situation where one wants to know whether a modeling activity (resulting in a theory with additional constraints on all kinds

\(^{18}\)We can also force it in the formula by stating \(P_i \cap \neg P_i \cap \cdots \cap \neg P_{i-1} \cap \neg P_{i+1}, \ldots, \cap \neg P_x\) for ever \(P_i(1 \leq i \leq x)\). This is not necessary to proof the reduction though, because we may without loss of correctness require satisfiable constraints on the models.
of information objects) is consistent. i.e. whether there are models (i.e. instances of information systems) that can satisfy all the constraints that were formulated during the modeling activity. This is a basic task for information processing.

Satisfiability for propositional logic is the archetypical case of satisfiability that is well known to be NP-complete. This means that for a formula \( A \) it takes 'exactly'\(^{19} \) a non-deterministic algorithm a polynomial number of steps (in terms of the length of \( A \)) to compute whether there is a model \( \mathcal{M} \) and an entity \( x \) in the model that satisfies \( A \).

This fact constrains the results of the analysis of the complexity of the logics of categorial graphs, because all of these logics\(^{20} \) contain propositional logic. This means that satisfiability for these logics for categorial graphs will at least have an NP-hard satisfiability task.

For the modal logics in general it is the case that most modal satisfaction tasks are not solvable in NP, but are at least PSPACE-hard. These tasks are solvable by a computational algorithm using only polynomial space. For example the minimal normal modal logic \( K \) and the modal logic \( S4 \) are PSPACE-complete. One way to get below PSPACE is when we can prove that the model that we need to construct to satisfy \( A \) is at most polynomial in size of \( A \). This polynomial size model property can be proved for models in which the constraints ensure that the model is compact\(^{21} \). Most models, however, can be used to simulate binary trees (i.e. exponential branching) and then this polynomial size model property fails and, moreover, proves PSPACE-hardness for the satisfiability task (cf. PSPACE-hardness criteria in [Spaan93]).

The situation sketched above implies some clear results for the individual logics for categorial graphs.

6.5.2. Theorem. The satisfiability task for the fragments of the adjacency logics where we have only one type of modality, either type 1 (\( \Diamond_1, \Box_1 \)) or type 2 (\( \Diamond_2, \Box_2 \)) are in NP.

Proof: This is directly implied by the constraint that the individual adjacency relations are at most only one step deep (forced by \( \Box \perp \)). This means that we

\(^{19}\) exactly' now means that is not easier than this; i.e. every task to which satisfiability of propositional formulas can be translated needs at least such an algorithm, and cannot be solved by an algorithm that is of lower computational complexity.

\(^{20}\) Actually all save one: when we take out the Boolean connectives, like in the example \( L_S \) we could get lower complexity.

\(^{21}\) When we have a polynomial size model property, we can let a non-deterministic algorithm guess a polynomial size model \( \mathcal{M} \) and entity \( x \). Now we need to do model checking for a polynomial number of times to verify that \( \mathcal{M} \) really is a model that satisfies the constraints that are put on the system (these constraints correspond to formula for our logics) and then do one other time model checking for \( A \) in \( \mathcal{M}, x \). Model checking is \( P \) in \( \mathcal{M} + |A| \), and thus also polynomial in \( |A| \). Hence we have an NP algorithm.
can branch over the individual $R_1$ relations only once, so we need at most $2 \times |A|$ entities to cover all the entities that $A$ can say something about.

6.5.3. THEOREM. *Satisfiability for the full adjacency modal logic is PSPACE-hard*

**Proof:** (Sketch) We can code the trees in two steps: a node is an $R_1$ source, an edge is an $R_2$ source, similar to where we presented the edge graphs formulation of a normal graph in chapter 3.

6.5.4. THEOREM. *Satisfiability for the extendibility logic is PSPACE-hard*

**Proof (Sketch):** The extendibility logic is, in its isolated shape, the logic S4.1 combined with the logic S4.2. The logic S4 is known to be PSPACE-hard, and this is proved by showing that its models can simulate trees\(^{22}\). The restriction of a local top of the McKinsey axiom (M) clearly does not frustrate this tree structure (we are certain to have a beautiful tree with one top).

An interesting corollary of the above result that requires a less sketchy proof, and exemplifies the fruits of the expressiveness of the logics for categorial graphs is the following: When we take a step further towards the combined logic add the $\diamond_1$ modality to the extendibility logic, we have a system in which we can define the 'tops' of the extendibility relations by $\diamond_1 \top$ (tops are the objects!). Now an S4 formula $A$ is satisfiable if and only if $A$ relativized\(^ {23} \) to $\neg \diamond_1 \top$ is satisfiable in the extendibility logic with the $\diamond_1$ modality. Hence the extendibility logic plus the $\diamond_1$ modality is PSPACE-hard.

The above results dash every hope to get the combined modal system in to NP, which coincides with the intuition that reasoning about complex objects is strictly more complex than simple propositional reasoning. On the other hand we have seen in the previous section that we do not have the burden of undecidability, that some models (like the standard associative models as we indicated above) have; and this implies that the satisfiability problem for the combined logic and its abstract models is also decidable. This shows the intriguing and surprising balance of languages and model classes. We gained the interesting insight here that reasoning about complex objects does not require full computational power of first order logic (i.e. undecidable). This is rather intuitive, but we note that in practice most reasoning algorithms are based on heuristics for full first order (undecidable) languages. Using the logics of categorial graphs as a basis would most likely give better results, because we then would build on a theory with better computational characteristics.

\(^{22}\)and therefore can do the well known PSPACE-complete Quantified Boolean Formula's task.

\(^{23}\)we say that a formula $A$ is relativized to a formula $\neg \diamond_1 \top$ when all modal subformula's of $A$ are replaced by $\diamond (\neg \diamond_1 \top \rightarrow B)$. This enables one to 'place' a top on the S4 model, such that it becomes a model for the extendibility logic plus the $\diamond_1$.
For the abstract resource models and substructural languages for categorial graphs we have a good case for the substructure adjacency fragment $L_{SA}$. These models have the finite model property for the substructural languages. The reason is the following: we can only talk resource consciously about the objects and their partial descriptions (which are entities in the model!), and thus all the specifications in the formula only have one entity in the model. Moreover, because this fragment does not contain negation, we do not need any combinatorics to construct a satisfying model. We can simply construct a satisfying model by introducing an entity for every building block of the formula: an atomic object for each propositional variable, an aggregate for each $A \ast B$ subformula, and a union of (sub) models for every conjunction $A \sqcap B$. Then we only need to check this one constructed model, because when it fails, then the formula cannot be satisfied by any model. This is evident by the lack of conflicting combinations in the valuation (only the relations may or may not satisfy the constraints of the logic, but the relations are fixed by the formula that we need to satisfy). We state the result without proof.

6.5.5. Theorem. The substructural language $L_{SA}$ with respect to the abstract resource models have a P-time satisfiability task.

The complexity most likely becomes less tractable when we do take into account the negation. The fragment with\(^{24}\) $\Diamond_1, \Diamond_2, \ast, \neg$ will still have the finite model property (hence $\text{NP}$ satisfiability), but the fragment including $\Diamond_1, \Diamond_2, \sqcap, \neg$ will not have the finite model property (hence $\text{PSPACE}$-hard satisfiability). The reason lies in whether or not the fragments are able to encode a binary tree shaped model.

6.5.6. Theorem. 1. The satisfiability task of the fragment of substructural $L_{FCA}$ with $\Diamond_1, \Diamond_2, \ast, \neg$ is $\text{NP}$

2. The satisfiability task of the fragment of substructural $L_{FCA}$ with $\Diamond_1, \Diamond_2, \sqcap, \neg$ is $\text{PSPACE}$-hard.

Proof: The first statement is proven by showing that every formula $A$ in the fragment can be satisfied by a model of maximal polynomial size in the length of $A$. We prove this by induction on the structure of $A$. It suffices to prove it for the $\ast, \neg$, and $\Diamond := \Diamond_1 \Diamond_2$, because only chains of alternating $\Diamond_1 \Diamond_2$ can be arbitrarily nested (remember the $\Box_i \Diamond_i \Box_j$ principle forcing each individual modality to be one step deep only). Let $\mathcal{M}, a \models A$, and $\mathcal{M} =< U, Q, R_1, R_2, V >$ with universe $U$, relations $Q, R_1, R_2$ and valuation $V$ as usual.

- suppose $A = P$ (where $P$ a proposition): take $\mathcal{M} =< \{a\}, \emptyset, \emptyset, \emptyset, V >$ with $V(P) = \{a\}$, and $V(P') = \emptyset$ for all propositions $P'$ other then $P$.

---

\(^{24}\)This fragment is obtained from $L_{SA}$ by adding negation but removing conjunction.
• Suppose $A = B_1 \ast B_2$: by induction we have polynomial size models $\mathcal{M}_1, \mathcal{M}_2$ such that $\mathcal{M}_1, b_1 \models B_1$ and $\mathcal{M}_2, b_2 \models B_2$. Now construct $\mathcal{M}$ by taking the union\(^{25}\) of (wlog supposed disjoint) models $\mathcal{M}_1, \mathcal{M}_2$ and extend $Q, V$ in $\mathcal{M}$ such that $aQb_1b_2$. We have now by definition of the interpretation that $\mathcal{M}, a \models A$ and size is simply $|\mathcal{M}_1| + |\mathcal{M}_2| + 1$.

• suppose $A = \Diamond B$. By induction we have a polynomial size model $\mathcal{M}'$ such that $\mathcal{M}', b \models B$. We extend $\mathcal{M}'$ to $\mathcal{M}$ by adding an object $a$ to the universe of $\mathcal{M}'$ and setting $aR_1R_2b$. We have now by definition of the interpretation that $\mathcal{M}, a \models A$ and size is simply augmented by a constant.

• suppose $A = \neg B$: By induction we have a polynomial size model $\mathcal{M}'$ such that $\mathcal{M}', b \models B$. We can transform $\mathcal{M}' = \langle U, Q, R_1, R_2, V' \rangle$ to $\mathcal{M} = \langle U, Q, R_1, R_2, V \rangle$ by altering the valuation as follows: For every entity $e$ in the universe of $\mathcal{M}'$ we invert the valuation with respect to the propositional variables that occur in $B$. This means that in $\mathcal{M}$ for all $P$ that occur in $B$ we put $V(P) = U - V'(P)$. It is clear that the size stays the same. We need to show that this (relatively cheap) construction works to ensure that $\mathcal{M}, a \models A$. This is seen by the following observation: For a model constructed in the process of this proof it holds that every subformula is local to one entity in the model, i.e. every modal subformula is satisfied by an atomic entity in $\mathcal{M}$ that has one $R_1R_2$ adjacent only (we do not have a conjunction to hold in the same entity, we only have aggregation that is satisfied by two independent entities). Spelling it out: Let $\bar{a}$ refer to an object in the inverted model $\mathcal{M}$ while $a$ refers to the object in model $\mathcal{M}'$. Now flipping the valuation to satisfy negation works for an object $a$ satisfying $P$, because $P$ will not hold in $\bar{a}$ anymore. For $B_1 \ast B_2$ that is satisfied in an aggregation object $a$ where $aQb_1b_2$, we also have that it does not hold in $\bar{a}$ anymore, because $B_1$ and $B_2$ do not hold anymore in their inverted subobjects $\bar{b}_1, \bar{b}_2$, and we are not allowed to misuse $\bar{b}_2$ to satisfy $B_2$ or misuse $\bar{b}_1$ to satisfy $B_1$, because we do not have exchange in the substructural calculus. Finally when $\Diamond B$ holds in $a$ because $B$ holds in $b$, where $aR_1R_2b$, then it will not hold anymore in $\bar{a}$, because there is only one $R_1R_2$ adjacent to $\bar{a}$, which is $\bar{b}$, and there $B$ is not satisfied.

We prove the second statement of the theorem by showing that we can force the model to have a tree shape, using a formula that is logarithmic in the size of the tree using the techniques developed by Hemaspaandra ([Spaan93]).

Let $q_0, \ldots, q_m$ and $p_1, \ldots, p_m$ be propositional variables. We will use the $q_i$ variables to encode the level in the tree\(^{26}\), and $p_i, \neg p_i$ to force the branching. We

---

\(^{25}\)Union of the universes and the relations.

\(^{26}\)Note we use objects as nodes in the tree and partial descriptions as edges. Thus we need to state $\Diamond_1\Diamond_2 p_i$ to force that $p_i$ holds one level deeper in the tree. The conditions $\Box_1 \Box_1 \perp$ and
Chapter 6. Logic of object oriented information

Figure 6.10: A model satisfying the branching formula \( B_0 \)

abbreviate a *branch* formula \( B_i \) as follows\(^{27}\):

\[
B_i := q_i \rightarrow (\Diamond_1 \Diamond_2(q_{i+1} \land p_{i+1}) \land \Diamond_1 \Diamond_2(q_{i+1} \land \neg p_{i+1}))
\]

Now we force branching\(^{28}\), by:

\[
\Box_1 \Box_2 B_i
\]

Now we need a formula that *sends* the truth values assigned to \( p_i \) and its negation one level down in the tree. This way we get the situation that once \( B_i \) has forced a branching in the model by creating a \( p_i \) and \( \neg p_i \) its truth values are sent down in the tree.

\[
S(p_i, \neg p_i) := (p_i \rightarrow \Box_1 \Box_2 p_i) \land (\neg p_i \rightarrow \Box_1 \Box_2 \neg p_i)
\]

to force a send we again use a necessation:

\[
\Box_1 \Box_2 S(p_i, \neg p_i)
\]

To force a tree of \( m \) levels we need to force \( m \) branchings; i.e. \( B_0 \land \Box_1 \Box_2 B_1 \land \Box_1 \Box_2 \Box_1 \Box_2 B_2 \land \cdots \land (\Box_1 \Box_2)^{m-1} \) and to force down the truth values all levels \( m \) we need the conjunction of \( m^2 \) formulas \((\Box_1 \Box_2)^j S(p_i, \neg p_i)\) (i.e. for all the \( m\) \( S(p_i, \neg p_i) \) we need then on on each level \( j \)).

Now we have a formula of size that is polynomial in the numbers of level of the tree it enforces; i.e. the formula forces a model that has exponential size with respect to its length. Hence we need at least \( \text{PSPACE} \) to compute satisfiability. \( \Box \)

For the concrete models we have the danger of the not-completely-axiomatizable system due to the richness of structure. Remember the undecidability result for the associative string calculus. In such case the satisfiability task is undecidable.

\(^{27}\)Standard branching formula is \( B_i := q_i \rightarrow (\Diamond(q_{i+1} \land p_{i+1}) \ast \Diamond(q_{i+1} \land \neg p_{i+1})) \).

\(^{28}\)In the standard encoding this is \( \Box B_i \)
6.5.4 Inference

Inference answers another, strongly related, question. It will compute whether a system of constraints from a modeling activity infers some other constraint. This task is the basis for verifying certain properties that are not explicitly forced by a system, but should follow from the theory and constraints that are actually implemented in a system.

For languages that have a (classical) negation connective and that have a complete calculus (w.r.t. their interpreting domain) the complexity results of the inference task and the satisfiability task are exactly the same. This is a consequence of the following statement:

\[ \vdash A \text{ if and only if } \neg A \text{ is not provable} \]

For characteristics of the inference task for the resource calculi we need to do some work. We have no negation in \( L_{SA} \), so we cannot use the results from the satisfiability task. Nevertheless the inference task is also expected to be \( P \)-time like the satisfiability task, due to the limited reasoning one can do in the language.

6.6 Extensions

In chapters 3 and 4 we briefly mentioned two extensions for the language of categorial graphs: the self and the ! (bang). These extensions are not part of the 'core' object intuition, but interesting extensions that enhance the ability to express constraints. In this section we will briefly discuss these extensions of the logic of categorial graphs.

In the object oriented paradigm, a statement (e.g. a constraint) is formulated from the point of view of an object. Formulating such a statement, it can be valuable to be able to refer to the object itself, i.e. to refer to the 'here and now' from the point of view where the statement is formulated. This can be accomplished by a modal constant self, which is interpreted to be true only in the evaluation point of the whole formula. One could then, for example, express that an object has itself as an adjacent:

\[ \Diamond_1 \Diamond_2 \text{self} \]

The self modality is not completely new. It is already studied in modern modal logic in the context of so-called hybrid languages ([BlackburnSeligman95]). The self modality fits into a nice extension of modal logic, where one has next to variables also so-called nominals. These nominals interpreted such that they are
true in exactly one object. The self modality is a special case. For this extension there exists a sound and complete axiomatization with respect to abstract relational semantics. Moreover, the complexity of the system is like most other modal systems: the satisfiability task is PSPACE-complete. We will not recite the axiomatics here. We argue that fruitful extensions from modal logic can be added\textsuperscript{29} to the core modal system of categorial graphs.

In the field of resource logics there is a well known modality that enables one to introduce the structural rules in a controlled manner: The ‘!’ (bang). The weakening, contraction, and exchange rules are introduced only for formulas labeled with the bang; e.g.

\[
\text{CONTRACTION} \quad \frac{\Gamma, !A, !A \Rightarrow \Delta}{\Gamma, !A \Rightarrow \Delta}
\]

The bang enables one to explicitly type an object that is an arbitrary long aggregation of objects of some kind. Structures like sets, multisets and lists are such kind of objects. In a setting like this (i.e. with the necessary bang-rules) an object that (itself) is a set of $A$-type objects can be typed as $!A$. In linear logic the bang is well studied. However, introducing the bang has quite some influence on the complexity of the system. The full propositional system (i.e. with $\ast, \cap, \cup, \neg$) with the bang for all the structural rules (together this is full propositional linear logic) is undecidable. From the computational point of view this modality should be introduced with care.

### 6.7 Further logical considerations

The main line of this chapter is a modal-substructural elaboration of our object-oriented information models. Its main ideas of adjacency and object-description duality also suggest other logical directions, however, even closer to classical first-order logic.

#### 6.7.1 'Object'/'type' duality

One example is the ‘object’/'type' duality found throughout standard logic. E.g., the information structures of Barwise & Seligman ([BarwiseSeligman97]) consist of sets of objects which can 'satisfy' or 'belong to' types, which one can think of as propositions, or sets in the extensional case. This is like our valuation $V$. The main logical structure imposed by these authors is the following:

\[
T \leq_{\text{type}} T' \text{ if every object satisfying } T \text{ also satisfies } T'
\]

\textsuperscript{29}Another nice, but less object-oriented flavored extensions worth looking at is the modal logic of inequality.
This is the usual implication ordering, and one may, or may not, require closure of the types under the Boolean operations. Dually, there is also an object inclusion:

\[ o \leq_{\text{object}} o' \text{ if every type } T \text{ that holds of } o \text{ also holds of } o' \]

This is like the 'specialization ordering' among points in topological spaces. Another close analogy is 'Chu Spaces', as studied extensively by Vaughan Pratt ([Benthem2000b]).

It is of some interest to compare this inclusion structure with our orderings: neither inclusion \( \leq \) is *exactly* what we had, though we could certainly define these additional relations.

\[ T \leq_{\text{type}} T' \text{ if } \diamond_1^uT \rightarrow \diamond_1^uT' \]

Note that we have a richer domain and need to take into account descriptions and objects, so simple implication ordering does not suffice. We need extendibility to make sure that descriptions that do not follow from each other, but accidentally describe the same (of a subset of each others) objects, are properly ordered. In information technology terms one could think about descriptions of the same objects from another perspective; e.g. the descriptions 'morning star' and 'evening star' do not 'include' each other but do describe the same object 'Venus'.

\[ o \leq_{\text{object}} o' \text{ if every description } d \text{ of } o \text{ there is a description } d' \text{ of } o' \text{ that is a witness of the same type of adjacent; i.e. } \forall d o R_1 d \exists d' o' R_1 d' (\exists a, a' dR_0 a c d' R_2 a' a' \models T \text{ iff } a' \models T) \]

Although this theory differs from ours, we can prove properties of it in our logics. Moreover our approach assumes that the objects themselves come with some prior structure, namely, the product construction. This richer structure, not found with Barwise & Seligman, then interacts with the inclusions: e.g., are products inclusion-monotone w.r.t. their components? This extension seems worth exploring.

### 6.7.2 Treating 'facts' as first-class citizens

But perhaps a still closer analogy to our view of information models lies right inside first-order predicate logic. The duality between objects and descriptions amounts to treating *facts* as first-class citizens in their own right, in the spirit of our discussion in Chapter 5. Our approach shows how one might do this in Tarski semantics. In addition to the ordinary universe of objects, take a second domain of 'descriptions', consisting of all the positive atomic facts that are true in the model. Facts can be of any arity, assigning properties to objects.

Normally the fact that, say, \( P \) holds between \( o \) and \( o' \) is modeled by putting the ordered pair \( < o, o' > \) in the set of pairs that forms the interpretation of \( P \). But
this 'reduction' is not necessary: we can say that $o, o'$ 'participate' in the fact in some more abstract - and yet more intuitive - way. Then it seems reasonable to say that our earlier relation $R_1$ holds between those objects and the fact "Poo". And $R_2$ is just its converse, linking a fact to the objects participating in it. Thus, our earlier analysis may be viewed as an analysis of the mutual ties between objects and facts, yielding an alternative ontology for predicate logic, and another locus for 'logical structure': $R_1, R_2$ are now basic *logical* items.

Now this would make $R_2$ just the converse of the relation $R_1$. This choice is natural. E.g., the composition $R_1; R_2$ will hold between any two objects that occur together in at least one positive fact of the model. This is the so-called 'Gaifman order' of a first-order model, which has various model-theoretic uses. Moreover, the idea of co-occurrence in a fact is precisely the main idea of the *guarded fragment* of predicate logic, an avant-garde development in modern modal logic ([AndrekaBenthemNemeti96], [RijkeVenema95]). Restricting quantification to guarded tuples of objects makes quantification 'local', and leads to decidability of the language.

On the other hand, our original intuition about object orientation was still a bit different. We were thinking of facts *about certain objects* as protagonists, with the others involved as auxiliary characters in the fact. This is why $R_1$ and $R_2$ are not inverses. This showed in our 'tagging' of facts to object: one atomic statement "Poo" could be two facts: one about $o$: viz. $< o, Poo' >$ and one about $o'$, viz. $< o', Poo' >$. This is another take on the same semantic setting - but we leave it to the logicians, or philosophers, to decide whether this additional 'aboutness' of facts is part of their essential structure.

Finally, *if* one reorganizes predicate-logical semantics in this way, then it makes sense to rethink the language as well. Should we not dualize everything, and allow quantification over facts? Our point with the present excursion is more modest, however. Far from being an exotic structure, object-description models in our sense might also be an interesting style of modeling basic logical structures, and one can think of our various logical systems then as axiomatizations for the 'mechanisms' that drive these new models.

To add some flesh to the considerations we describe three modal logic versions of the above idea:

- **version A**: We take a domain of objects $O$ and a domain of facts *about adjacency* between objects $R_{Adj}$. Between these two domains we have two relations $\pi_1, \pi_2$ that relate an adjacency fact to, respectively, the object and its adjacent

- **version B**: We generalize the domain of facts and allow *arbitrary facts* $T(a), U(a, b), V(a, b, c) \ldots$ about the objects in $O$. We have a relation $\pi$ between the facts on the objects, relating a fact to and object when the object occurs in the fact.
Further logical considerations

Version C: one domain of entities consists now of sets of basic objects that occur in facts; and similarly we take the domain with sets of facts (hey, we are taking things together like in the object oriented calculus!). A set of facts \( Y \) relates to a set of objects \( X \) when at least one of the objects in \( X \) occurs in at least one of the facts in \( Y \).

Version A is actually the adjacency logic where \( R_1 \) is the converse of \( \pi_1 \) and \( R_2 \) coincides with \( \pi_2 \). The axiomatization therefore could be similar. However, from this point of view a symmetric set of axioms for modalities for \( \pi_1 \) and \( \pi_2 \) seems more natural. Let \( < \pi_1 >, < \pi_2 > \) be the modalities interpreted by \( \pi_1, \pi_2 \) respectively, and let \([\pi_1], [\pi_2]\) be their dual modalities and \( < \pi_1 >^\cup, < \pi_2 >^\cup \) their inverses. To force proper behavior of the \( \pi_i \) relations in the light of objects and adjacency facts we then typically get (next to the normal modal principles) the following principles in the logic for adjacency facts:

(Disjoint) \( \neg ((< \pi_1 > \land < \pi_2 > \land T) \land (< \pi_1 >^\cup \land < \pi_2 >^\cup \land T)) \)
(Exhaustive) \( (< \pi_1 > \lor < \pi_2 > \lor T) \lor (< \pi_1 >^\uparrow \lor < \pi_2 >^\uparrow \lor T)) \)
\( (\pi_1 \text{ is a function}) \quad < \pi_1 > A \rightarrow [\pi_1]A \)
\( (\pi_2 \text{ is a function}) \quad < \pi_2 > A \rightarrow [\pi_2]A \)
\( (\text{First Order}) \quad [\pi_1][\pi_1] \perp \equiv [\pi_2][\pi_2] \perp \)

Note that we cannot express that when two facts have their projections to the same objects we actually are talking about the same fact; i.e. if

\[ f_{\pi_1 o_1 \& f_{\pi_2 o_2} \& f'_{\pi_1 o_1 \& f'_{\pi_2 o_2}} \]

then \( f = f' \). Although this uniqueness is an important property of facts, we can ignore the matter using this logic for analyzing these facts, because all general models for our logic of adjacency facts are bi-similar to a special model that does have this uniqueness property.
Version B reveals interesting principles for a full predicate model with facts as first-class citizens in modal terms. Let $R$ be the relation between objects and facts where $oRf$ if object $o$ occurs in fact $f$ (e.g. $f = T(o)$). Let $\diamond_1$ be the modal operator that is interpreted by $R$ with its dual $\Box_1$, its inverse $\check{\diamond}_1$, and its inverses dual $\check{\Box}_1$. The logic for the first-class-facts typically looks as follows:\(^{30}\)

\begin{align*}
&(\text{Disjoint}) \quad \neg(\check{\diamond}_1 T \cap \diamond_1 T) \\
&(\text{Exhaustive}) \quad \diamond_1 T \cup \diamond_1 T \\
&(\text{First order}) \quad \Box_1 \Box_1 \bot
\end{align*}

This system forces every object to play a role in a fact (i.e. there are no uninteresting objects where we know nothing of). An alternative that loosens this constraint (and introduces some asymmetry between facts and objects) has the following alternative rules:

\begin{align*}
&(\text{Disjoint'}) \quad \neg(\check{\diamond}_1 \bot \cap \diamond_1 T) \\
&(\text{Exhaustive'}) \quad \diamond_1 \bot \cup \diamond_1 T \\
\end{align*}

Version C translates our intuition on aggregating descriptions and objects to an intuition on aggregating facts and objects. Let $R$ be the relation between sets of objects and sets of facts where $oR_{\mathcal{F}}f$ if some object $o \in O$ occurs in some fact $f \in F$. Let $\diamond_{\mathcal{U}}$ be the modal operator that is interpreted by $R$ with its dual $\Box_{\mathcal{U}}$, its inverse $\check{\diamond}_{\mathcal{U}}$, and its inverses dual $\check{\Box}_{\mathcal{U}}$. Also let $\cup_1, \cup_2, \cup_3$ be the versatile triple that models an abstract relation\(^{31}\) for union $\cup$; i.e. $ZUXY$ relates $Z$ to $X$ and $Y$ when $Z$ is the union of $X$ and $Y$. Now we can say some things about monotonicity of the relation between sets of objects and sets of facts. The logic for the sets of first class facts typically has next to the normal modal principles for unary and dyadic modal operators the following principles:

\begin{align*}
&(\text{Disjoint}) \quad \neg(\diamond_{\mathcal{U}} T \cap \diamond_{\mathcal{F}} T) \\
&(\text{Exhaustive}) \quad \diamond_{\mathcal{U}} T \cup \diamond_{\mathcal{F}} T \\
&(\text{First order}) \quad \Box_{\mathcal{U}} \Box_{\mathcal{F}} \bot \\
&(\text{Downward monotonicity}) \quad (X \cap \diamond_{\mathcal{U}} F_1) \cup_1 Y \rightarrow \diamond_{\mathcal{U}} F_1 \\
&(\text{Upward monotonicity}) \quad (F_1 \cap \diamond_{\mathcal{F}} X) \cup_1 F_2 \rightarrow \diamond_{\mathcal{F}} X
\end{align*}

In the combined system of categorial graphs we have seen more complex monotonicity principles: the regularity principles. For these principles we had stronger versions too. The stronger equivalents in this logic relate 'aggregating' sets of objects to 'aggregating' sets of facts (cf. the regularity axioms for the combined system for categorial graphs in definition 6.2.19):

\begin{align*}
&(\text{Downward regularity}) \quad (X \cap \diamond_{\mathcal{U}} F_1) \cup_1 (Y \cap \diamond_{\mathcal{U}} F_2) \rightarrow \diamond_{\mathcal{U}} (F_1 \cup_1 F_2) \\
&(\text{Upward regularity}) \quad (F_1 \cap \diamond_{\mathcal{F}} X) \cup_1 (F_2 \cap \diamond_{\mathcal{F}} Y) \rightarrow \diamond_{\mathcal{F}} (X \cup_1 Y)
\end{align*}

\(^{30}\)Note that we can leave out the principle for requiring a source and targets for the facts

\(^{31}\)We can not force all the axioms of set theory by modal principles for the set-union relation.
What we have seen here is that intuitions on how to talk about object oriented structures can be mapped on the setting of relational structures that were invented to interpret first order predicate logic. This seems promising for the intuition for the categorial graphs when we realize that the language of categorial graphs was intended to talk about the real world (modeled in object oriented structures) while first order predicate logic in turn is a language that intends to talk about phenomena in the real world (and is interpreted in relational structures).

Moreover, it shows that our intuitions on object orientation could possibly have consequences for standard logic, because it introduces a new view on objects and facts. This view could be beneficial to the field of logic itself.

### 6.8 Summary

In this chapter we analyzed the concepts of the system of categorial graphs of the previous chapters from a perspective of modern logic. We have disclosed logical properties of the individual core notions and have given account of their axiomatics and complexity, using results from the field of modal and substructural logic. The results provided us with a clear view on the core concepts of object orientation. On the other hand we provided the logicians with a concrete system that bears some interesting logical questions. Moreover, we proposed a different view on logic itself, using an intuition from object oriented practice. We strongly believe this scientific cross-fertilization bears even more fruits that are be beneficial for both the computer modeling field and logic. Here lies a challenge to be taken on in further research.