An Invitation to Game Comonads

Tomáš Jakl

Luca Reggio

Department of Computer Science and Technology, University of Cambridge, UK
Email address: tomas.jakl@cl.cam.ac.uk
URL: https://tomas.jakl.one

Department of Computer Science, University College London, UK
Email address: l.reggio@ucl.ac.uk
URL: https://lucareggio.github.io

Draft version compiled on 7th August 2022.

Contents

Note to the reader

Generally speaking, a certain degree of mathematical maturity is expected from the students. This includes familiarity with basic notions of discrete mathematics, such as sets, functions, relations, partial orders, and mathematical induction. Familiarity with more advanced topics in order theory, such as Galois connections between posets, would be useful but is not essential as the relevant notions will be recalled in the course. With regard to logic and category theory, the prerequisites are detailed below:

Logic: Basic familiarity with (classical) propositional and first-order logics is assumed, as well as with the concepts of syntax and semantics, as can be found in any textbook on the subject (a self-contained, albeit terse, account can be found in §2.1 of (Libkin, 2004)). This includes, in particular, the ability to read a formula in first-order logic and interpret it in a model. A similar degree of acquaintance with modal logic (corresponding roughly to §§1.1–1.3 of (Blackburn et al., 2001)) is desirable, although the relevant concepts will be reviewed in Chapter 1.

Category theory: Familiarity with basic notions of category theory is a prerequisite for the course. These include the concepts of categories, functors, natural transformations, isomorphisms, monics, and epics. This material (and much more) can be found in the first two chapters of (Adámek et al., 1990), or in §§1.1–1.5 of (Abramsky and Tzevelekos, 2011). Some familiarity with more advanced notions, such as adjunctions, limits, colimits, universal constructions, and comonads, is useful but not essential: these concepts will be motivated and introduced during the course through the examples of game comonads.

Introduction

These lecture notes are written for the course *Relating Structure to Power: An Invitation to Game Comonads*, taught by the two authors at the 33rd European Summer School in Logic, in Galway (Ireland) in 2022. Our main aim is to provide the first beginner friendly introduction to the study of game comonads.

The study of game comonads originates in the study of two problems in computer science which are famously computationally difficult. The first one is the so-called *Constraint Satisfaction Problem* (CSP). It can be presented as a problem of deciding if, for a fixed (testing) relational structure B, there exists a homomorphism $A \to B$, for a structure A given on input. Similarly, the *isomorphism problem* is a decision problem determining if A and B are isomorphic, that is, $A \cong B$.

Both problems have a logical reading. When A and B are finite, it is known that $A \cong B$ precisely when A and B satisfy the same formulas in first-order logic or in the *first-order logic with counting quantifiers*. Similarly, the existence of a homomorphism $A \to B$ amounts to checking if B satisfies *primitive positive* formulas which are true in A.

The difficulty of both problems let the research communities to the study their approximations. From the point of view of logic, this means restricting the involved formulas in both above problems. In particular, instead of studying the equivalence of A and B in (full) first-order logic, we aim to decide a weaker equivalence, that is, a logical equivalence in a selected fragment of first-order logic (with or without counting quantifiers). Similarly, instead of checking if B satisfies all primitive positive formulas that hold in A, we only check those that satisfy certain syntactic requirements.

It turns out that many such approximations can be decided in polynomial time. Moreover, these approximations can be often presented as a game between Spoiler, who is trying to show that either $A \not\to B$ or $A \ncong B$, and Duplicator who is defending the opposite position.

Game comonads come from the idea that these games can be formalised purely semantically. In fact, Duplicator winning strategies in the approximation games of $A \to B$ correspond to homomorphisms $A \to_G B$ in the Kleisli category for some comonad G. Similar encodings which make use of the same comonad G can be made for the game-theoretic approximations of the isomorphism problems too. This encoding of games brings an important twist to the study of these problems and potentially also to model theory as

well. With game comonads we can reason about logic fragments and their expressivity abstractly, by using the language of category theory. Moreover, the study of the formally dual notion, that is, the study of monads has a very long tradition in category theory. We make use of the rich literature on the subject, with an eye on applications in finite model theory.

One example of the borrowing from the theory of monads is when we turn our attention to coalgebras of our comonads. Since its beginnings, the theory of monads concentrated on the study of algebras of monads, which provides a generalisation of universal algebra to settings where possibly more structure is present; for example, to the study of topological groups. The formal dual of algebras for a monad are coalgebras of a comonad. Perhaps surprisingly, coalgebras of our game comonads correspond to some well-known decompositions from combinatorics, such as forest cover and tree decompositions and thereby also tree-depth and tree-width graph parameters.

There is a tight connection between logic fragments and combinatorial parameters in the study CSP and isomorphism problems. The theory of game comonads makes this connection explicit. Furthermore, by formalising our proofs categorically, with the said comonad as a parameter, we are able to obtain many results in a uniform fashion. The well-known problems are recovered by instantiations of the comonad in question.

Although the study of game comonads started quite recently, in 2017, it has already shown a lot of potential. Many logic fragments and combinatorial parameters have been captured and some of the most important theorems from finite model theory have been formalised in the language of comonads. However, there is still a whole avenue of interesting research directions, both from the point of view of category theory and finite model theory, that awaits more attention. We hope that this text provides enough inspiration for such explorations.

Warning: These notes are currently under heavy development. We are only providing a rough draft of the first chapter and also the necessary categorical preliminaries at the moment. The other listed chapters are in various stages of writing up and will be released in due time. Admittedly, some examples, exercises and references are to be added. Moreover, the current version of this text does not include any results related to modal logic and modal comonads.

Guided bibliography

Game comonads represent a recent advance in relating categorical semantics to finite model theory, and their study has gathered considerable momentum. As such, there is no single reference for their basic theory, which has been developed in a series of research articles. To guide the reader through the emerging theory of game comonads, we indicate some references relevant to the course, for those who might wish to explore some of the topics more fully.

For a gentle introduction to the fundamental ideas of game comonads, we warmly recommend the following survey:

[1] Abramsky, *Whither semantics?*, Theoretical Computer Science, Vol. 807, 2020. Preprint available at https://arxiv.org/abs/2010.13328

The core of the course, focussing on game comonads, is based on the following research articles (in particular, we recommend [4] as the gateway to the basic results on game comonads):

[2] Abramsky, Dawar, Wang, *The pebbling comonad in finite model theory*, in the proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LiCS), 2017. Preprint available at https://arxiv.org/abs/1704.05124

[3] Abramsky, Shah, *Relating Structure and Power: Comonadic semantics for computational resources*, in the proceedings of the 27th EACSL Annual Conference on Computer Science Logic (CSL), 2018. Freely available at https://drops.dagstuhl.de/opus/volltexte/2018/9669

[4] Abramsky, Shah, *Relating Structure and Power: Extended version*, Journal of Logic and Computation, 31(6):1390–1428, 2021. Preprint available at https://arxiv.org/abs/2010.06496

[5] Dawar, Jakl, Reggio, *Lovász-type theorems and game comonads*, in the proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LiCS), 2021. Preprint available at https://arxiv.org/abs/2105.03274

The following more advanced articles on game comonads are also relevant, and at the end of the course the students will have the necessary background to undertake self-study:

[6] Abramsky, Marsden, *Comonadic semantics for guarded fragments*, in the proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LiCS), 2021. Preprint available at https://arxiv.org/abs/2008.11094

[7] Abramsky, Reggio, *Arboreal categories and resources*, in the proceedings of the 48th International Colloquium on Automata, Languages and Programming (ICALP), 2021. Freely available at https://drops.dagstuhl.de/opus/volltexte/2021/14184/

[8] Ó Conghaile, Dawar, *Game comonads & generalised quantifiers*, in the proceedings of the 29th EACSL Annual Conference on Computer Science Logic (CSL), 2021. Freely available at https://drops.dagstuhl.de/opus/volltexte/2021/13450

[9] Paine, *A pebbling comonad for finite rank and variable logic, and an application to the equirank-variable homomorphism preservation theorem*, Electronic Notes in Theoretical Computer Science, 352:191–209, 2020. Freely available at https://doi.org/10.1016/j.entcs.2020.09.010

The following monographs on category theory, finite model theory, and modal logic, are meant to guide the interested reader who may wish to learn more about these topics. Familiarity with these texts is not a prerequisite for the course (except for some basic concepts of logic and category theory, as detailed in the *Note to the Reader*).

[10] Adámek, Herrlich, Strecker, *Abstract and concrete categories, The joy of cats*, Pure and Applied Mathematics, John Wiley & Sons, Inc., New York, 1990. xiv+482 pp. Online edition freely available at http://katmat.math.uni-bremen.de/acc/acc.pdf

[11] Abramsky, Tzevelekos, *Introduction to categories and categorical logic*, New Structures for Physics, B. Coecke (ed). Lecture Notes in Physics Vol. 813, pp. 3–94, Springer-Verlag, 2011. Preprint available at https://arxiv.org/abs/1102.1313

[12] Libkin, *Elements of finite model theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2004. xiv+315 pp. Freely available at https://homepages.inf.ed.ac.uk/libkin/fmt/fmt.pdf

[13] Blackburn, de Rijke, Venema, *Modal logic*, Cambridge Tracts in Theoretical Computer Science, Vol. 53, Cambridge University Press, Cambridge, 2001. xxii+554 pp.

CHAPTER 1

Syntax and semantics: a fruitful duality

In this chapter we explain the basics of the interplay between syntax and semantics in logic. This relationship is well-known in model theory. We however look at it from the point of view of finite model theory. Concretely, we explain the Chandra–Merlin correspondence which is a refinement of the standard model theoretic correspondence between syntax and semantics to finite structures. As we will see in later chapters, this correspondence plays a fundamental role in the theory of game comonads and neatly explains why some of the constructions explained later work the way they do. We also look at refinements of the Chandra–Merlin correspondence to the restricted variable and restricted quantifier-rank fragments of logics.

1.1. Theories and structures

Syntax and semantics: Correspondences between theories and sets of structures, the Chandra-Merlin correspondence between finite structures and primitive positive sentences. The homomorphism order and homomorphism equivalence.

In the CSP and isomorphism problems, we can only compare structures of the same 'type'. This is done by specifying a *relational signature* σ, that is, specifying a (finite) set of names of relation symbols $\{R_1, \ldots, R_n\}$ together with their arities. The arity $\mathtt{ar}(R)$ of a relation symbol R in σ is a positive integer. Typically, instead of writing down the arities explicitly, we specify them implicitly, *e.g.* by writing $\sigma = \{R(\cdot,\cdot),\, P(\cdot),\, S(\cdot,\cdot,\cdot)\}$ to mean that σ consists of a binary relation R, unary relation P and a ternary relation S.

A relational signature σ defines the *category of σ-structures* $\mathbf{Str}(\sigma)$. Every σ-structure A is specified by its universe A (*i.e.* a set of *vertices* or *points*) and an interpretation of every relation in the signature. That is, for a relation symbol $R \in \sigma$ of arity n we have a set of n-tuples (also called *edges* or *relations*)

$$R^A \subseteq A^n.$$

Homomorphisms of σ-structures are required to preserve these relations. Concretely, for σ-structures A, B, a homomorphism $f\colon A \to B$ is a function between the universes such that, for every $(a_1, \ldots, a_n)$ in R^A, the tuple $(f(a_1), \ldots, f(a_n))$ is in R^B.

Further, upon fixing a relational signature σ, we may specify a fragment of first-order logic that allows us to talk about properties of σ-structures. As usual we form formulas in terms of existential and universal quantifiers $\exists, \forall$,

conjunctions and disjunctions $\wedge, \vee$, negations $\neg$ and atomic formulas. As atomic formulas we allow equality $x = y$ between variables and an n-ary predicate $R(x_1, \ldots, x_n)$ for every n-ary relation symbol R in the signature σ. We also include the always true statement $\mathbf{t}$ and always false $\mathbf{f}$ statements among atoms. In the following we write $\mathrm{FO}(\sigma)$ for the set of all first-order *sentences* in signature σ.

Example 1.1. The sentence $\exists x.\mathbf{t}$ (in any signature σ) expresses the fact that a σ-structure is non-empty. Further, for the relational signature of directed graphs σ, that is, $\sigma = \{E(\cdot,\cdot)\}$, the first-order sentence

$$\varphi(x,y) \stackrel{\text{def}}{\equiv} \exists z_1(E(x,z_1) \wedge \exists z_2(E(z_1,z_2) \wedge E(z_2,y)))$$

expresses that there is a path of length 3 between x and y. ▲

In the following we make use of two standard relations on σ-structures. We say that σ-structures A, B are *logically equivalent* in a fragment Ł of first-order logic, and write

$$A \equiv^{\mathcal{L}} B,$$

if, for every sentence $\varphi \in \mathcal{L}$, $A \vDash \varphi$ holds precisely whenever $B \vDash \varphi$ holds. Furthermore, we write

$$A \Rrightarrow^{\mathcal{L}} B,$$

whenever, for every sentence $\varphi \in \mathcal{L}$, if $A \vDash \varphi$ then also $B \vDash \varphi$.

It is an important feature of finite structures that $A \cong B$ iff $A \equiv^{\mathrm{FO}(\sigma)} B$. This is made more precise in the following exercise.

Exercise 1.2. Let A be a finite σ-structure with universe $\{a_1, \ldots, a_n\}$ and write γ_A for the *'graph'* of the structure, that is the sentence

$$\exists x_1, \ldots, x_n \ (\varphi_{pos} \wedge \varphi_{neg} \wedge \forall y\,(y = x_1 \vee \cdots \vee y = x_n))$$

where φ_{pos} is the formula in free variables $x_1, \ldots, x_n$ obtained as the conjunction of all $R(x_{k_1}, \ldots, x_{k_u})$ such that $(a_{k_1}, \ldots, a_{k_u}) \in R^A$, for some $R \in \sigma$. Similarly, and ψ_{neg} is the conjunction of all $\neg R(x_{k_1}, \ldots, x_{k_u})$ such that $(a_{k_1}, \ldots, a_{k_u}) \notin R^A$.

Given a σ-structure B, show the following.

(a) $A \cong B$ iff $B \vDash \gamma_A$,
(b) $A \cong B$ iff $A \equiv^{\mathcal{L}} B$, for every logic $\mathcal{L}$ such that $\gamma_A \in \mathcal{L}$, and
(c) $A \cong B$ iff $A \equiv^{\mathrm{FO}(\sigma)} B$. ▲

Exercise 1.3. Show that whenever a fragment of logic $\mathcal{L}$ is closed under negation (*i.e.* $\varphi \in \mathcal{L}$ implies $\neg\varphi \leftrightarrow \psi$ for some $\psi \in \mathcal{L}$) then $A \equiv^{\mathcal{L}} B$ is equivalent to $A \Rrightarrow^{\mathcal{L}} B$. In particular, $A \equiv^{\mathrm{FO}(\sigma)} B$ holds precisely whenever $A \Rrightarrow^{\mathrm{FO}(\sigma)} B$ does. ▲

Exercise 1.4. Let σ be the signature of directed graphs, $\sigma = \{E(\cdot,\cdot)\}$. If $\mathcal{L}$ is the fragment of first-order sentences of the form

$$\exists x_1, \ldots, x_n \ \varphi(x_1, \ldots, x_n)$$

where φ is a conjunction of atomic formulas, then an infinite path with a starting vertex A and a single vertex with a loop B, shown in Figure 1 below, satisfy that $A \Rrightarrow^{\mathcal{L}} B$. Find the sentence that proves that the converse is not true. ▲

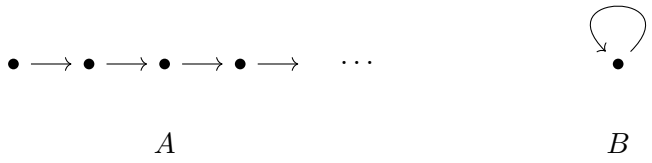


FIGURE 1. Two directed graphs

1.1.1. Model-theoretic correspondences. It is a standard fact that, by Gödel's completeness theorem, that a theory is consistent if and only if it has a model. Furthermore, there is a one-to-one correspondence between

(1) complete consistent theories T, and
(2) non-empty classes of σ-structures Δ such that $A \in \Delta$ if and only if $A \equiv^{\mathrm{FO}(\sigma)} B$ for every $B \in \Delta$.

The correspondence is established by assigning to a complete consistent theory T the class of its models $\Delta = \{A \mid A \vDash T\}$. Conversely, given a class Δ as in (2), the theory $\mathbf{Th}(A) = \{\varphi \in \mathrm{FO}(\sigma) \mid A \vDash \varphi\}$ of an arbitrary $A \in \Delta$ is complete and consistent.

In fact, first-order logic (in general) does not distinguish non-isomorphic structures. Whenever a theory has an infinite model then it has many more different non-isomorphic models. In particular, even if a theory is so-called ω-categorical (*i.e.* has only one countable model, up to isomorphism), by Löwenheim–Skolem Theorem, it has a model of every infinite cardinality.

This changes when we focus on finite structures, that is, we study the subcategory $\mathbf{Str}_{fin}(\sigma)$ of $\mathbf{Str}(\sigma)$ consisting of σ-structures with finite universe. Then, logical equivalence already captures isomorphism, *cf.* Exercise 1.2. In fact, the theory of a single finite structure is determined by one sentence, as shown in the following exercise.

Exercise 1.5. For a finite σ-structure A and sentence $\varphi \in \mathrm{FO}(\sigma)$, show that $\varphi \in \mathbf{Th}(A)$ iff γ_A implies φ, where γ_A is the graph of A from Exercise 1.2. ▲

It is a consequence of Löwenheim–Skolem Theorem that a theory has a unique model (up-to isomorphism) if and only if it has a unique finite model. Therefore, the (1)–(2) correspondence above restricts to finite structures as follows. There is a one-to-one correspondence between

(1') sentences in $\mathrm{FO}(\sigma)$ (up-to equivalence) with a unique model, and
(2') finite σ-structures (up-to isomorphism).

Now comes a twist. The (1')–(2') correspondence only allows us to describe structures up-to isomorphism which, as we know, is difficult to verify.

By the example of Chandra and Merlin (1977) we look at a variation of this correspondence and restrict the sentences in the logic to *primitive positive sentences*. These are the first-order sentences in signature σ where disjunction, universal quantification and negation is not allowed. For simplicity, we also do not consider sentences with equality. Put differently, we only allow sentences obtained as a combination of existential quantification $\exists$, conjunctions $\wedge$, and also $R(x_1, \ldots, x_n)$, for every n-ary relation symbol R in the language σ. Moreover, we also allow the always true statement $\mathbf{t}$. Write $\mathrm{PP}(\sigma)$ for the collection of all primitive positive sentences in signature σ.

Example 1.6. Note that all sentences appearing in Examples 1.1 and 1.4 are primitive positive. ▲

It is an important feature of this logic that the primitive positive theory

$$\mathbf{Th}_{\mathrm{PP}(\sigma)}(A) = \{\varphi \in \mathrm{PP}(\sigma) \mid A \vDash \varphi\}$$

of a finite structure A is uniquely determined by a single primitive positive sentence. Namely, let

$$\Psi \colon \mathbf{Str}_{fin}(\sigma) \to \mathrm{PP}(\sigma)$$

be the operation that assigns to a finite σ-structure A on universe $\{a_1, \ldots, a_n\}$ its *primitive positive 'graph'* $\Psi(A)$, defined as follows:

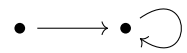
$$\exists x_1 \ldots \exists x_n \bigwedge_{R \in \sigma} \bigwedge_{(a_{i_1}, \ldots, a_{i_r}) \in R^A} R(x_{i_1}, \ldots, x_{i_r}) \tag{1.1}$$

Observe that if we only care about sentences up-to interprovability, by the usual rules of first-order logic, re-ordering of elements of the structure will not change the resulting sentence.

As we show later, in Proposition 1.10 below, $\Psi(A)$ encodes everything that is true of A in $\mathrm{PP}(\sigma)$, *i.e.* for a primitive positive sentence φ,

$$\varphi \in \mathbf{Th}_{\mathrm{PP}(\sigma)}(A) \iff \Psi(A) \vdash \varphi. \tag{1.2}$$

Example 1.7. Let σ be the signature of graphs and let A be the graph on two vertices with a loop, as drawn below.



Then, $\Psi(A)$ is the sentence $\exists x_1 x_2 . E(x_1, x_2) \wedge E(x_2, x_2)$. Notice that, the main difference between $\Psi(A)$ and the sentence γ_A, defined in Exercise 1.2, is that $\Psi(A)$ omits the "negative" part of the sentence as well as the part that says that every vertex is either x_1 or x_2. Indeed, γ_A is equivalent to the following sentence.

$$\begin{aligned}\exists x_1 x_2 \, (&E(x_1, x_2) \wedge E(x_2, x_2) \wedge \neg E(x_2, x_1) \wedge \neg E(x_1, x_1) \\ &\wedge \forall y \, (y = x_1 \vee y = x_2))\end{aligned}$$

▲

The property (1.2) of Ψ above establishes that the primitive positive theory of a finite structure is uniquely described by a single sentence. In fact, every primitive positive sentence can be obtained from a 'unique' finite σ-structure. Although, this time the uniqueness is not up-to isomorphism but instead up-to homomorphism-equivalence, as we explain later on.

In order to establish a correspondence between primitive positive sentences and finite structures, we need a mapping in the opposite direction. Define

$$\mathbf{M}\colon \mathrm{PP}(\sigma) \to \mathbf{Str}_{fin}(\sigma)$$

as the operation that builds a finite σ-structure $\mathbf{M}(\varphi)$ out of a primitive positive sentence φ as follows. First, we assume that no two quantifiers bind the same variable name in φ (if this is not the case, we can rename the variables first). Further, recall that formulas $\exists y\,(\varphi \wedge \psi)$ and $(\exists y\,\varphi) \wedge \psi$ are equivalent in first-order logic if y does not occur freely in ψ. Consequently, φ is equivalent to a sentence in the *prenex normal form*

$$\exists x_1 \,\ldots\, \exists x_n \; \psi(x_1, \ldots, x_n)$$

where ψ is just a conjunction of atoms. We therefore build $\mathbf{M}(\varphi)$ as the structure on the set $\{x_1, \ldots, x_n\}$, with relations between the elements precisely as given by ψ.

Let us now record essential properties of the two operations. Given σ-structures A, B, in the following we write

$$A \to B$$

to express the fact that *there exists a homomorphism from A to B.*

Lemma 1.8. *Let A be a finite σ-structure and φ a primitive positive sentence. Then:*

(i) $A \vDash \Psi(A)$
(ii) $\mathbf{M}(\varphi) \vDash \varphi$
(iii) $\mathbf{M}(\Psi(A)) \cong A$

Furthermore, for a σ-structure B of arbitrary size,

(iv) $B \vDash \varphi$ *iff* $\mathbf{M}(\varphi) \to B$.

Proof. Item (iii) follows directly from the definitions and items (i) and (ii) follow directly from (iii) and (iv). Next, we carefully check (iv). Let $\exists x_1, \ldots, x_n\, \psi(x_1, \ldots, x_n)$ be the prenex normal of φ, where ψ is a conjunction of atoms.

If $B \vDash \psi$ then, by definition, there exist $b_1, \ldots, b_n \in B$ such that $B \vDash \psi(b_1, \ldots, b_n)$. We define a function

$$f\colon \mathbf{M}(\varphi) \to B$$

as the mapping $x_i \mapsto b_i$. Observe that for a tuple $(x_{i_1}, \ldots, x_{i_k})$ in $R^{\mathbf{M}(\varphi)}$, we have that $B \vDash R(b_{i_1}, \ldots, b_{i_k})$ since $B \vDash \psi(b_1, \ldots, b_n)$. In other words, for

$(x_{i_1}, \ldots, x_{i_k})$ in $R^{\mathbf{M}(\varphi)}$ implies that $(f(x_{i_1}), \ldots, f(x_{i_k}))$ is in R^B or, in other words, f is a σ-structure homomorphism.

Conversely, given a homomorphism $f\colon \mathbf{M}(\varphi) \to B$, we wish to prove that $B \vDash \varphi$. It is enough to show that, for the assignment

$$b_1 = f(x_1), \ \ldots, \ b_n = f(x_n),$$

we have that $B \vDash \psi(b_1, \ldots, b_n)$. But this follows from the fact that f is a homomorphism. Indeed, for any atom $R(x_{i_1}, \ldots, x_{i_k})$ appearing in the conjunction ψ, the tuple $(x_{i_1}, \ldots, x_{i_k})$ belongs to $R^{\mathbf{M}(\varphi)}$. Then, since f is a homomorphism, the tuple $(f(x_{i_1}), \ldots, f(x_{i_k})) = (b_{i_1}, \ldots, b_{i_k})$ is in R^B or, in other words, $B \vDash R(b_{i_1}, \ldots, b_{i_k})$. Consequently, we obtain $B \vDash \psi(b_1, \ldots, b_n)$ since the choice of atoms from the conjunction was arbitrary. □

This careful and concrete examination now pays off. We can establish further important properties of the two operations, simply by an abstract manipulations from (i)–(iv).

Lemma 1.9. *For σ-structures A, B and primitive positive sentences φ, ψ,*

(v) If $A \to B$ and $A \vDash \varphi$ then $B \vDash \varphi$,
(vi) $A \to B$ iff $B \vDash \Psi(A)$ (assuming A is finite), and
(vii) $\varphi \vdash \psi$ implies $\mathbf{M}(\psi) \to \mathbf{M}(\varphi)$.

Proof. (v) By (iv), $A \vDash \varphi$ is equivalent to $\mathbf{M}(\varphi) \to A$. Then, by composition with $A \to B$ we obtain a homomorphism $\mathbf{M}(\varphi) \to B$. Therefore, by (iv) again, $B \vDash \varphi$.

(vi) The left-to-right direction follows directly from (i) and (v) as $A \vDash \Psi(A)$ and $A \to B$ implies $B \vDash \Psi(A)$. Conversely, by (iv), $B \vDash \Psi(A)$ implies $\mathbf{M}(\Psi(A)) \to B$. Then, by (iii), we see that $A \to B$.

(vii) By (ii) we see that $\mathbf{M}(\varphi) \vDash \psi$ and then (iv) gives $\mathbf{M}(\psi) \to \mathbf{M}(\varphi)$. □

With the previous two lemmas we can now address the promised equivalence from (1.2).

Proposition 1.10. *Let A be a finite structure and $\varphi \in \mathrm{PP}(\sigma)$, then*

$$A \vDash \varphi \iff \Psi(A) \vdash \varphi.$$

Proof. Assume that $A \vDash \varphi$ and assume further that $B \vDash \Psi(A)$ for some σ-structure B. The former is equivalent to $\mathbf{M}(\varphi) \to A$ by (iv) and the latter to $A \to B$ by (vi). By homomorphism composition we obtain $\mathbf{M}(\varphi) \to B$ which by (iv) makes $B \vDash \varphi$. We have showed that any structure satisfying $\Psi(A)$ also satisfies φ. By completeness, that gives that $\Psi(A) \vdash \varphi$.

Conversely, assume $\Psi(A) \vdash \varphi$. Then, by (vii), $\mathbf{M}(\varphi) \to \mathbf{M}(\Psi(A))$. However, then (iii) entails $\mathbf{M}(\varphi) \to A$. Consequently, by (iv), $A \vDash \varphi$. □

Coming back to the classical model-theoretic correspondence between (1) and (2), let us examine what does the restriction to primitive positive sentences, on one hand, and finite structures, on the other, buys us.

We have observed that the primitive positive theory $\mathbf{Th}_{\mathrm{PP}(\sigma)}(A)$ of a finite structure is determined by a single primitive positive sentence, that is, its primitive positive 'graph' $\Psi(A)$. Therefore, instead of dealing with theories, as in (1), we may restrict to individual sentences. Conversely, from (v) it follows, that the class $\Delta = \{A \in \mathbf{Str}_{fin}(\sigma) \mid A \vDash \varphi\}$ of models of a single sentence $\varphi \in \mathrm{PP}(\sigma)$ is upwards closed in the *homomorphism order*, *i.e.* if $A \in \Delta$ and $A \to B$ then also $B \in \Delta$. Furthermore, Δ is completely determined by the sentence $\mathbf{M}(\varphi)$, which also belongs to Δ. Indeed, by (iv), $B \in \Delta$ iff $\mathbf{M}(\varphi) \to B$.

To obtain a correspondence similar to (1') and (2'), we observe that the choices of the "determining sentence" $\Psi(A)$ of a primitive positive theory of A and the "determining structure" $\mathbf{M}(\varphi)$ for the class of models of a primitive positive sentence φ, are unique up-to a reasonable notion of equivalence. As before, the reasonable notion of equivalence for sentences is the interprovability (that is, when $\varphi \vdash \psi$ and $\psi \vdash \varphi$) which is the same as that φ and ψ are equivalent in the logic (*i.e.* $\vdash \varphi \leftrightarrow \psi$). For structures A, B, the reasonable notion of equivalence is the *homomorphism equivalence*

$$A \leftrightarrows B$$

i.e. the existence of a homomorphism $A \to B$ as well as $B \to A$.

Observe that $\mathbf{M}$ and Ψ are invariant under these equivalences. Indeed, by (vii), if $\varphi \leftrightarrow \psi$ then $\mathbf{M}(\varphi) \leftrightarrows \mathbf{M}(\psi)$ and, conversely, $A \leftrightarrows B$ implies $\Psi(A) \leftrightarrow \Psi(B)$ by the following lemma.

Lemma 1.11. *For finite σ-structures A, B,*

(viii) $A \to B$ *implies* $\Psi(B) \vdash \Psi(A)$.

Proof. $A \to B$ implies $B \vDash \Psi(A)$ by (vi) and then by Proposition 1.10 also $\Psi(B) \vdash \Psi(A)$. □

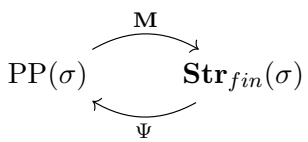
In the following we observe that homomorphism equivalence and $\equiv^{\mathrm{PP}(\sigma)}$-equivalence are the same relation on finite structures. In fact, we show that $A \Rrightarrow^{\mathrm{PP}(\sigma)} B$ is is characterised by the homomorphism order.

Proposition 1.12. *Given a finite A and arbitrary B,*

$$A \to B \quad \textit{iff} \quad A \Rrightarrow^{\mathrm{PP}(\sigma)} B.$$

Proof. Observe that (v) already expresses the left-to-right direction. For the converse direction, observe that $A \vDash \Psi(A)$ by (i) and so, by our assumption $A \Rrightarrow^{\mathrm{PP}(\sigma)} B$, also $B \vDash \Psi(A)$. Therefore, by (vi), $A \to B$. □

To summarise, the *Chandra-Merlin correspondence* is given by the two mappings Ψ and $\mathbf{M}$ between $\mathrm{PP}(\sigma)$ and the class $\mathbf{Str}_{fin}(\sigma)$ of finite relational structures in signature σ.



As we have shown, one of the characterising features of this correspondence is that we can transfer between syntactic and semantic characterisations of the satisfaction relation. That is, for A finite and $\varphi \in \mathrm{PP}(\sigma)$:

$$\mathbf{M}(\varphi) \to A \iff A \models \varphi \iff \Psi(A) \vdash \varphi \tag{1.3}$$

Moreover, $\mathbf{M}$ and Ψ establish a one-to-one equivalence between

(1") sentences in $\mathrm{PP}(\sigma)$ (up-to equivalence), and
(2") finite σ-structures (up-to homomorphism equivalence).

Exercise 1.13. Provide the missing details for the proof of (i)–(iii) in Lemma 1.8. Hint: Use (iv) to show (ii) and (iii)+(iv) for (i). ▲

1.1.2. The order-theoretic point of view. We might view $\mathrm{PP}(\sigma)$ and $\mathbf{Str}_{fin}(\sigma)$ as preorders. The set of primitive positive sentences $\mathrm{PP}(\sigma)$ may be ordered by $\vdash$, that is, for sentences $\varphi, \psi \in \mathrm{PP}(\sigma)$ we say that φ is less than or equal to ψ iff $\varphi \vdash \psi$. Similarly, we might order $\mathbf{Str}_{fin}(\sigma)$ by the homomorphism order, that is, for finite structures A, B, we say that A is less than or equal to B iff there exists a homomorphism $A \to B$.

Under these order, (vii) and (viii) simply say that Ψ and $\mathbf{M}$ are antitone mappings between the two preorders $\mathrm{PP}(\sigma)$ and $\mathbf{Str}_{fin}(\sigma)$. In fact, $\mathbf{M}$ and Ψ can be viewed as the maps witnessing that $\mathrm{PP}(\sigma)$ and $\mathbf{Str}_{fin}(\sigma)$ are isomorphic, as preorders. By (iii) we know that any finite structure A is isomorphic to $\mathbf{M}(\Psi(A))$. The following shows the corresponding fact for the composition in the opposite order.

Lemma 1.14. *For any primitive positive sentence φ,*

(ix) $\vdash \Psi(\mathbf{M}(\varphi)) \leftrightarrow \varphi$

Proof. By completeness $\varphi \vdash \Psi(\mathbf{M}(\varphi))$ is equivalent to the statement that, for every σ-structure A,

$$A \models \varphi \text{ implies } A \models \Psi(\mathbf{M}(\varphi)).$$

To show that, let $A \models \varphi$. By (1.3), this implies $\mathbf{M}(\varphi) \to A$. Since Ψ is antitone by (viii), we have that $\Psi(A) \vdash \Psi(\mathbf{M}(\varphi))$. Finally, by (1.3) again, $A \models \Psi(\mathbf{M}(\varphi))$.

The converse direction follows by Proposition 1.10 and (iii). □

Exercise 1.15. The preorders $\mathrm{PP}(\sigma)$ and $\mathbf{Str}_{fin}(\sigma)$ are not posets. Find two different sentences φ, ψ which are equivalent $\vdash \varphi \leftrightarrow \psi$ and, similarly, find two non-isomorphic finite structures A, B which are homomorphically equivalent, that is, $A \leftrightarrows B$. ▲

1.2. Resource-bounded logics

finite-variable and bounded quantifier-rank fragments, their interplay with the Chandra-Merlin correspondence, modal logic

As we said earlier, establishing whether there exists a homomorphism $A \to B$ or whether A and B are isomorphic $A \cong B$ are very difficult computational problems. Establishing some computational feasibility requires us to look at at approximations of these problems. Let us look at the approximations to the homomorphism problem first.

One approach of finding approximations to the homomorphism problem is offered by the link made in Proposition 1.12. Observe that for any fragment Ł of $\mathrm{PP}(\sigma)$ the relation $\Rrightarrow^{\text{Ł}}$ is coarser than $\Rrightarrow^{\mathrm{PP}(\sigma)}$ and, more importantly, that gives us that

$$A \to B \quad \text{implies} \quad A \Rrightarrow^{\mathcal{L}} B.$$

In other words, for any fragment $\mathcal{L}$ of $\mathrm{PP}(\sigma)$, the relation $\Rrightarrow^{\mathcal{L}}$ is an approximation of the homomorphism order. In fact, the above implication is probably more natural in its negative form, which is more suitable for application:

$$A \not\Rrightarrow^{\mathcal{L}} B \quad \text{implies} \quad A \not\to B.$$

If we can decide $\Rrightarrow^{\mathcal{L}}$ quickly then we can quickly refute some instances where $A \to B$ does not hold. Ideally we want the relation $\Rrightarrow^{\mathcal{L}}$ to be as close to the homomorphism order $\to$ as possible, while still maintaining good computational properties. Moreover, there are well-known important cases where $\Rrightarrow^{\mathcal{L}}$ exactly matches $\to$ for a carefully chosen $\mathcal{L}$, see *e.g.* Barto and Kozik (2014).

Probably the most natural way to find fragments of $\mathrm{PP}(\sigma)$ is by having syntactic restriction on sentences of $\mathrm{PP}(\sigma)$. Recall that primitive positive formulas are formed from atoms, in terms of conjunctions $\wedge$ and existential quantification $\exists$. Naturally, we can impose restrictions on what types of conjunctions we allow or, similarly, what types of existential quantifiers we allow. The former is found for example in the work of Dalmau (2005).

In the following we look at two types of restrictions of quantifications which have been widely studied in the literature and for which it is known that the relation $\Rrightarrow^{\mathcal{L}}$ can be computed by an efficient algorithm.

1.2.1. Bounded quantifier rank fragments. Probably the most natural syntactic restriction is the one where we bound the depth of nesting of quantifiers. Formally, this is done by defining the *quantifier rank* $\mathtt{qrank}(\varphi)$ of a formula φ as a function from formulas to natural numbers. We define it inductively, on the structure of primitive positive sentences by:

$$\begin{aligned} \mathtt{qrank}(A) &:= 0 \qquad\qquad (\text{for an atomic formula } A) \\ \mathtt{qrank}(\varphi \wedge \psi) &:= \max(\mathtt{qrank}(\varphi), \mathtt{qrank}(\psi)) \\ \mathtt{qrank}(\exists x\, \varphi) &:= \mathtt{qrank}(\varphi) + 1 \end{aligned}$$

In fact, this definition can be naturally extended to arbitrary formulas of first-order logic as follows.

$$\begin{aligned}\texttt{qrank}(\neg\varphi) &:= \texttt{qrank}(\varphi)\\ \texttt{qrank}(\varphi \vee \psi) &:= \max(\texttt{qrank}(\varphi), \texttt{qrank}(\psi))\\ \texttt{qrank}(\forall x\, \varphi) &:= \texttt{qrank}(\varphi) + 1\end{aligned}$$

Example 1.16. Let us consider the following sentence

$$\varphi = \exists xy\,(R(x,y) \wedge R(y,x) \wedge \exists z\, S(x,y,z)).$$

Since this is our first example where we consider syntactic restrictions, we are going to be pedantic and write the sentence φ carefully, with all brackets that are typically dropped by convention:

$$\exists x\,(\exists y\,(R(x,y) \wedge (R(y,x) \wedge \exists z S(x,y,z)))).$$

Next we show how that the quantifier rank $\texttt{qrank}(\varphi)$ of φ is equal to 3, by applying the inductive definition of $\texttt{qrank}(\cdot)$ step by step.

$$\begin{aligned}\texttt{qrank}(\varphi) &= \texttt{qrank}(\exists x\,(\exists y\,(R(x,y) \wedge (R(y,x) \wedge \exists z S(x,y,z)))))\\ &= \texttt{qrank}(\exists y\,(R(x,y) \wedge (R(y,x) \wedge \exists z S(x,y,z)))) + 1\\ &= \texttt{qrank}(R(x,y) \wedge (R(y,x) \wedge \exists z S(x,y,z))) + 2\\ &= \max(\texttt{qrank}(R(x,y)), \texttt{qrank}(R(y,x) \wedge \exists z S(x,y,z))) + 2\\ &= \max(0, \max(\texttt{qrank}(R(y,x)), \texttt{qrank}(\exists z S(x,y,z)))) + 2\\ &= \max(0, \max(0, \texttt{qrank}(S(x,y,z)) + 1)) + 2\\ &= \max(0, \max(0,1)) + 2\\ &= 1 + 2 = 3\end{aligned}$$

Observe that the order of conjunctions does not actually matter because $\texttt{qrank}(\cdot)$ is defined as the maximum of quantifier ranks of the two parts, which is a commutative operation. This shows very important later on in Section 1.2.3 because it allows us to encode the nesting of quantifiers as tree orders on structures. ▲

For a natural number k, denote by $\mathrm{FO}_k(\sigma)$ the *quantifier-rank k* fragment of first-order logic, which consists of the sentences in first-order logic $\varphi \in \mathrm{FO}(\sigma)$ such that $\texttt{qrank}(\varphi) \leq k$. Similarly, we define $\mathrm{PP}_k(\sigma)$ as the restriction of $\mathrm{PP}(\sigma)$ to sentences in $\mathrm{FO}_k(\sigma)$.

Since $\mathrm{PP}_k(\sigma)$ is a fragment of $\mathrm{PP}(\sigma)$, Proposition 1.12 gives us that

$$A \to B \quad \text{implies} \quad A \Rrightarrow^{\mathrm{PP}_k(\sigma)} B.$$

Furthermore, $\mathrm{FO}_k(\sigma)$ is a fragment of $\mathrm{FO}(\sigma)$ and so, by Exercise 1.2(c), we see that the logical equivalence in $\mathrm{FO}_k(\sigma)$ approximates isomorphism. Indeed, for finite structures A, B, we have that

$$A \cong B \quad \text{implies} \quad A \equiv^{\mathrm{FO}_k(\sigma)} B.$$

We conclude this part by noting an important fact about this fragment.

Proposition 1.17. *Up-to equivalence there are only finitely many sentences in* $\mathrm{FO}_k(\sigma)$.

Consequently, there are also only finitely many sentences in $\mathrm{PP}_k(\sigma)$.

Example 1.18. Observe that $\mathrm{FO}_k(\sigma)$ is not closed under equivalence. There exist sentences in first-order logic φ, ψ such that $\varphi \notin \mathrm{FO}_k(\sigma)$ but $\vdash \varphi \leftrightarrow \psi$. In fact, the same can be said about the $\mathrm{PP}_k(\sigma)$ fragment. Construct such sentences. ▲

Exercise 1.19 (advanced)**.** Prove Proposition 1.17. Hint: It is important that we only allowed finite signatures which are finitary, that is, the relation symbols in our language are of finite arity. ▲

1.2.2. Bounded variable fragments. Another possible syntactic restriction is by allowing only certain variables to be used in the quantifications. For every natural number k we fix a set of pairwise different variables $\{x_1, \ldots, x_k\}$. Then, we write $\mathrm{FO}^k(\sigma)$ for *k-variable* fragment of first-order logic, that is, for the collection of first-order sentences that only use variables among $x_1, \ldots, x_k$. Similarly, we write $\mathrm{PP}^k(\sigma)$ for k-variable primitive positive sentences, *i.e.* for the collection of sentences in the intersection of $\mathrm{PP}(\sigma)$ and $\mathrm{FO}^k(\sigma)$. Note that we do not restrict the quantifier rank of sentences in these two fragments.

For example, the following is a sentence in $\mathrm{PP}^2(\sigma)$

$$\exists x_1 \left(\exists x_2 \left(R(x_1, x_2) \wedge \exists x_1 \left(R(x_2, x_1) \wedge R(x_1, x_1)\right)\right)\right). \tag{1.4}$$

On the other hand, the following is not in $\mathrm{PP}^2(\sigma)$, even though, the two sentences are equivalent.

$$\exists x_1 \left(\exists x_2 \left(R(x_1, x_2) \wedge \exists x_3 \left(R(x_2, x_3) \wedge R(x_3, x_3)\right)\right)\right)$$

It turns out that the $\mathrm{FO}^k(\sigma)$ and $\mathrm{PP}^k(\sigma)$ fragments are already quite expressive. In fact, $\mathrm{FO}^k(\sigma)$ is more expressive than the fragment $\mathrm{FO}_k(\sigma)$ we defined in the previous section since, for any A, B, we have that

$$A \equiv^{\mathrm{FO}^k(\sigma)} B \quad \text{implies} \quad A \equiv^{\mathrm{FO}_k(\sigma)} B$$

and also that

$$A \Rrightarrow^{\mathrm{PP}^k(\sigma)} B \quad \text{implies} \quad A \Rrightarrow^{\mathrm{PP}_k(\sigma)} B.$$

These important facts follow from the following lemma.

Lemma 1.20. *Given a sentence φ in $\mathrm{FO}_k(\sigma)$, there is a sentence ψ in $\mathrm{FO}^k(\sigma)$, of the same quantifier rank, such that φ and ψ are equivalent.*

Moreover, the same statement holds when restricting to sentences in $\mathrm{PP}_k(\sigma)$ and $\mathrm{PP}^k(\sigma)$, respectively.

Proof Idea. We rename the variables in φ to $x_1, \ldots, x_n$ based on their nesting depth. The out-most quantifications will bind x_1, all next-level quantifications bind x_2, and so on.

For example, the sentence $\exists x(\exists y\, R(x, y) \wedge \exists z\, R(z, x))$ gets renamed to the sentence $\exists x_1(\exists x_2\, R(x_1, x_2) \wedge \exists x_2\, R(x_2, x_1))$. □

The proof of Lemma 1.20 demonstrates that, from the point of view of expressibility of $\mathrm{FO}^k(\sigma)$, it is enough to look at formulas up-to variable renaming. Although, we have restricted sentences to only those that use the variables among $x_1, \ldots, x_n$ in $\mathrm{FO}^k(\sigma)$, morally, in this fragment we can express any property that is expressible in a sentence that uses at most k distinct variables, whichever they are. For example, the sentence in (1.4) can be more conveniently written as:

$$\exists x \left(\exists y \left(R(x,y) \wedge \exists x \left(R(y,x) \wedge R(x,x)\right)\right)\right)$$

Indeed, since we only used variables x, y, we can do the renaming $x \mapsto x_1$ and $y \mapsto x_2$, to obtain an equivalent sentence in $\mathrm{FO}^2(\sigma)$.

Example 1.21. A statement similar to Proposition 1.17 cannot be proved for the k-variable fragment. Find an infinite set of mutually non-equivalent sentences in $\mathrm{FO}^k(\sigma)$ for $k = 2$.

Hint: In the language of graphs, you can say in $\mathrm{FO}^2(\sigma)$ that there exists a path of length n, for every natural number n. ▲

1.2.3. Tree order via the Chandra–Merlin correspondence. In the following we would like to understand how the syntactic restrictions we made in Sections 1.2.1 and 1.2.2 translate as properties of finite structures via the Chandra–Merlin correspondence.

Observe that in the construction of $\mathbf{M}(\varphi)$, for a primitive positive sentence φ, as we defined it in Section 1.1.1, we throw away all information about syntactic properties of φ that were important to us in the previous two sections. More specifically, in the definition of $\mathbf{M}(\varphi)$ we first rename all variables in φ so that no variable is quantified twice and then we convert the formula to its prenex normal form, which throws away the order of quantifier nesting.

In the following, we show that $\mathbf{M}(\varphi)$ can be equipped with a compatible forest order that correspond to the syntactic properties of φ, before we did any syntactic modifications to it. Furthermore, this extra tree order can be used to characterise σ-structures that arise from sentences in $\mathrm{FO}_k(\sigma)$ and $\mathrm{FO}^k(\sigma)$, respectively.

Before we define the tree order on $\mathbf{M}(\varphi)$ we make a notational distinction between the different formulas appearing in the construction, starting from a primitive positive sentence φ:

- Denote by φ_V the sentence obtained from φ after the renaming of variables so that no variable is quantified twice.
- We assume that the variables occurring in φ_V after this renaming are exactly $v_1, \ldots, v_n$.
- Denote by
$$\exists v_1, \ldots, v_n \, \varphi_A(v_1, \ldots, v_n)$$
the prenex normal form of φ_V, where φ_A is just a conjunction of atoms.

With this, we are now ready to define the tree order on $\mathbf{M}(\varphi)$. Recall that $\mathbf{M}(\varphi)$ is a σ-structure with the universe $\{v_1, \dots, v_n\}$. For elements v_i, v_j of $\mathbf{M}(\varphi)$, write

$$v_i \leq v_j$$

if v_j appears in the scope of variable v_i in φ_V.

Lemma 1.22. *Given a sentence $\varphi \in \mathrm{PP}(\sigma)$ and $A = \mathbf{M}(\varphi)$, the binary relation $\leq$ on A, defined as above is a forest order:*

- *(T1) $\leq$ is a partial order, that is, $\leq$ is reflexive, transitive and anti-symmetric.*
- *(T2) For every element $a \in A$, the set $\downarrow a = \{x \in A \mid x \leq a\}$ is a finite set, linearly ordered by $\leq$.*

Furthermore, it satisfies the following compatibility condition:

- *(T3) If $(a_1, \dots, a_n) \in R^A$ then, for every $i, j \in \{1, \dots, n\}$, either*

$$a_i \leq a_j \qquad \text{or} \qquad a_j \leq a_i.$$

Proof. (T1) and (T2) follow directly from the fact that formulas in first-order logic are tree-shaped. To see why (T3) holds it is enough to realise that $(v_1, \dots, v_n) \in R^{\mathbf{M}(\varphi)}$ is only true if the atom $R(v_1, \dots, v_n)$ appears somewhere in the formula φ_V. Then, since φ_V is a sentence, the variables $v_1, \dots, v_n$ are all quantified. Let us assume that v_i was quantified as the last. But then we have that $v_1, \dots, v_n \in \downarrow v_i$. Consequently, these variables, seen as elements of $\mathbf{M}(\varphi)$, are pairwise $\leq$-comparable. □

Given a σ-structure A and a binary relation $\leq$ on A, we say that $\leq$ is a *compatible forest order on A* if it satisfies conditions (T1)–(T3). If, furthermore, every linear chain $\downarrow a$ in A has size $\leq k$, we say that the forest order *has depth k*. Finally, denote by $\mathcal{F}_k(\sigma)$ the class of finite structures $A \in \mathbf{Str}_{fin}(\sigma)$ which admit a compatible forest order of depth k.

With this notation, we see that Lemma 1.22 implies that $\mathbf{M}$ restricts to a mapping

$$\mathbf{M} \colon \mathrm{PP}_k(\sigma) \to \mathcal{F}_k(\sigma).$$

It turns out that axioms (T1)–(T3) and the requirement on depth characterises the σ-structures under the $\mathbf{M}$ image of sentences in $\mathrm{PP}_k(\sigma)$. In fact, one can also show that Ψ restricts to a mapping between $\mathcal{F}_k(\sigma)$ and $\mathrm{PP}_k(\sigma)$ as well. The only caveat to this is that we have to consider formulas up-to equivalence in logic. The following theorem expresses that the restrictions of Ψ and $\mathbf{M}$ are still inverse to each other, if we look at elements of $\mathrm{PP}_k(\sigma)$ and $\mathcal{F}_k(\sigma)$ up-to their natural equivalence.

Theorem 1.23. *For every finite σ-structure A, the following are equivalent:*

1. *$A \in \mathcal{F}_k(\sigma)$.*
2. *A is isomorphic to $\mathbf{M}(\varphi)$ for some $\varphi \in \mathrm{PP}_k(\sigma)$.*

And, similarly, for every primitive positive sentence φ, the following are equivalent:

(3) φ is equivalent to $\Psi(A)$ for some $A \in \mathcal{F}_k(\sigma)$.
(4) φ is equivalent to a sentence $\psi \in \mathrm{PP}_k(\sigma)$.

PROOF. We start by showing a stronger statement than the implication "(3) $\Rightarrow$ (4)". We show that, for a structure A,

($\star$) if $A \in \mathcal{F}_k(\sigma)$, then $\Psi(A)$ is equivalent to some $\psi \in \mathrm{PP}_k(\sigma)$ such that $A \cong \mathbf{M}(\psi)$.

Let $\leq$ be the compatible forest order of depth k, witnessing $A \in \mathcal{F}_k(\sigma)$. We assume that the universe $\{a_1, \ldots, a_n\}$ of A is enumerated by a depth-first search exploration of vertices of A according to the $\leq$-order. Concretely, we start the depth-first search from a root (*i.e.* minimal element) and enumerate elements following any branch all the way to a leaf (*i.e.* a maximal element). Once a leaf is reached, we backtrack to the last visited vertex which branches and continue enumerating from there to a leaf. We repeat this procedure until we enumerated the whole tree of our starting root. After that, we move to the next root and continue the same way and so on, until we have enumerated the whole structure. There might be multiple enumerations of the universe, we pick an arbitrary one. Since

$$\exists x \exists y\, \varphi \quad \text{and} \quad \exists y \exists x\, \varphi$$

are equivalent formulas, the sentence $\Psi(A)$ is equivalent to the same sentence with the quantifier prenex reordered according to the enumeration of the universe of A:

$$\exists v_1, \ldots, v_n\ \varphi_A(v_1, \ldots, v_n) \tag{1.5}$$

where φ_A is the conjunction of $R(v_{i_1}, \ldots, v_{i_r})$ such that $(a_{i_1}, \ldots, a_{i_r}) \in R^A$.

Next, we aim to find a formula equivalent to (1.5), which has quantifier rank k. We wish to transform the formula to the shape that mirrors the order $\leq$ on A. Recall that conjunctions distribute over existential quantifiers for subformulas that are not bound by the quantifier. This means that if the variable y does not occur freely in ψ then the formulas

$$\exists y\, (\varphi \wedge \psi) \quad \text{and} \quad (\exists y\, \varphi) \wedge \psi$$

are equivalent, according to rules and axioms of first-order logic.

Going backwards in the depth-first search enumeration of A, we transform (1.5) to an equivalent formula, distributing conjunctions over existential quantifiers according to the above restriction. In particular, assuming φ is a conjunct where variable v_{i+1} does not occur and ψ is a conjunct where it does, we transform

$$\exists v_i \exists v_{i+1}\ (\varphi \wedge \psi)$$

to

$$\exists v_i\, (\varphi \wedge (\exists v_{i+1}\, \psi))$$

whenever $a_i \leq a_{i+1}$ in A, and to

$$(\exists v_i\, \varphi) \wedge (\exists v_{i+1} \psi)$$

whenever $a_i \not\leq a_{i+1}$ in A.

The quantifiers in the resulting formula γ are ordered according to the $\leq$-order on A, that is, $a_i \leq a_j$ if and only if v_j is in the scope of quantification of v_i. Correctness of this procedure follows from (T4) for $\leq$. Consequently, $\mathtt{qrank}(\gamma) \leq k$ because $\leq$ is of depth k.

Furthermore, it is clear that the resulting formula γ has the same number of quantifiers as there are elements of the universe of A and the atomic formulas appearing in the formula correspond precisely to tuples in relations of A. Therefore, $A \cong \mathbf{M}(\gamma)$.

We can now justify the claimed equivalences. We have shown that (2) implies (1) in Lemma 1.22. The converse as well as "(3) $\Rightarrow$ (4)" follow directly from ($\star$). For "(4) $\Rightarrow$ (3)", assume that $\varphi \leftrightarrow \psi$ for some $\psi \in \mathrm{PP}_k(\sigma)$. Observe that

$$\varphi \leftrightarrow \psi \leftrightarrow \Psi(\mathbf{M}(\psi))$$

where the second equivalence holds by (ix). We have shown that φ is equivalent to $\Psi(A)$ with $A = \mathbf{M}(\psi)$, which is in $\mathcal{F}_k(\sigma)$ by "(2) $\Rightarrow$ (1)". □

As a corollary, we obtain that the relation $\Rrightarrow^{\mathrm{PP}_k(\sigma)}$ can be characterised purely semantically, akin to the characterisation of $\Rrightarrow^{\mathrm{PP}(\sigma)}$ in terms of the homomorphism order in Proposition 1.12.

Corollary 1.24. *For σ-structures A, B,*

$$A \Rrightarrow^{\mathrm{PP}_k(\sigma)} B \iff \forall C \in \mathcal{F}_k(\sigma)\ C \to A \text{ implies } C \to B$$

Example 1.25. Using Proposition 1.17, show that there are only finitely many structures in $\mathcal{F}_k(\sigma)$, up-to homomorphism equivalence. ▲

1.2.4. Pebble function via the Chandra–Merlin correspondence. Here we describe the restriction of the correspondence between finite structures and primitive positive sentences to $\mathrm{PP}^k(\sigma)$. Let us first recall the notation from the beginning of Section 1.2.3. For a sentence $\varphi \in \mathrm{PP}(\sigma)$, we denote by φ_V the sentence obtained by renaming variables in φ and then the sentence

$$\exists v_1, \ldots, v_n\ \varphi_A(v_1, \ldots, v_n).$$

is taken to be the prenex normal form of φ_V, on the same set of variables.

Given a k-variable sentence $\varphi \in \mathrm{PP}^k(\sigma)$, Lemma 1.22 still ensures that we can equip $\mathbf{M}(\varphi)$ with a compatible forest order. However, this time the trees can have arbitrary depth. In order to describe the σ-structures in the image of $\mathrm{PP}^k(\sigma)$ under the mapping $\mathbf{M}$ we need to provide further information about the usage of variables in the original formula. To this end, we define a *pebble function*

$$p\colon \mathbf{M}(\varphi) \to \{1, \ldots, k\}$$

by sending v_i to the number $n \in \{1, \dots, k\}$ which represents the index of variable x_n, from the set of allowed variables $x_1, \dots, x_k$ in $\mathrm{PP}^k(\sigma)$, such that x_n got renamed to v_i in the process of normalising φ to φ_V.

Next we observe that a similar property to (T3) above is guaranteed, explaining the interplay between the forest order and the pebble function. The proof is analogous to the proof of Lemma 1.22. The main difference is that, this time, we also keep track of which quantifications overshadow each other. We leave the proof to the reader.

Lemma 1.26. *Given a sentence $\varphi \in \mathrm{PP}^k(\sigma)$ and $A = \mathbf{M}(\varphi)$ with the compatible forest order $\leq$ as in Lemma 1.22, the pebble function $p\colon A \to \{1, \dots, p\}$ defined as above satisfies:*

(T4) If $(a_1, \dots, a_n) \in R^A$ then, for every $i, j \in \{1, \dots, n\}$, either

- *(a) $a_i \leq a_j$ and $p(a_i) \neq p(b)$ for every $b \in A$ such that $a_i < b \leq a_j$, or*
- *(b) $a_j \leq a_i$ and $p(a_j) \neq p(b)$ for every $b \in A$ such that $a_j < b \leq a_i$.*

Given a σ-structure A, a binary relation $\leq$ on A and a function $p\colon A \to \{1, \dots, n\}$, we say that $\leq, p$ is a *compatible k-pebble forest order on A* if it satisfies conditions (T1), (T2), and (T4). Denote by $\mathcal{F}_k(\sigma)$ the class of finite structures $A \in \mathbf{Str}_{fin}(\sigma)$ which admit a compatible k-pebble forest order.

Observe that (T4) is a stronger condition than (T3) is. Therefore, any compatible k-pebble forest order is a compatible forest order, in sense of Section 1.2.3.

As before, Lemma 1.26 implies that $\mathbf{M}$ restrict to a mapping of the following type.

$$\mathbf{M}\colon \mathrm{PP}^k(\sigma) \to \mathcal{F}^k(\sigma)$$

We also have a theorem similar to Theorem 1.23, whose proof we leave as an exercise to the reader.

Theorem 1.27. *For every finite σ-structure A, the following are equivalent:*

- *$A \in \mathcal{F}^k(\sigma)$.*
- *A is isomorphic to $\mathbf{M}(\varphi)$ for some $\varphi \in \mathrm{PP}^k(\sigma)$.*

And, similarly, for every primitive positive sentence φ, the following are equivalent:

- *φ is equivalent to $\Psi(A)$ for some $A \in \mathcal{F}^k(\sigma)$.*
- *φ is equivalent to a sentence $\psi \in \mathrm{PP}^k(\sigma)$.*

Finally, from the last theorem it directly follows that we have the following semantic representation of the $\Rrightarrow^{\mathrm{PP}^k(\sigma)}$ relation.

Corollary 1.28. *For σ-structures A, B,*

$$A \Rrightarrow^{\mathrm{PP}^k(\sigma)} B \iff \forall C \in \mathcal{F}^k(\sigma)\ C \to A \text{ implies } C \to B$$

Exercise 1.29. Adapt the proof of Theorem 1.23 to show Theorem 1.27. ▲

Exercise 1.30. Define $\mathrm{FO}_n^k(\sigma)$ as the intersection of $\mathrm{FO}^k(\sigma)$ and $\mathrm{FO}_n(\sigma)$ and define $\mathcal{F}_n^k(\sigma)$ as the collection of finite σ-structures which admit a compatible k-pebble forest order of depth n.

Adapt the statements in the last two sections to obtain that, for σ-structures A, B,

$$A \Rrightarrow^{\mathrm{PP}_n^k(\sigma)} B \iff \forall C \in \mathcal{F}_n^k(\sigma)\ C \to A \text{ implies } C \to B.$$ ▲

1.3. Fragments of modal logic

modal signature, pointed Kripke structures, modal formulas, bounded modal-depth

This section is currently being written and will be added in due time.

CHAPTER 2

Games and game comonads

This chapter is yet to be written.

CHAPTER 3

Coalgebras and combinatorial parameters

This chapter is yet to be written.

3.1. Forest covers and tree decompositions

3.2. Eilenberg–Moore coalgebras

3.3. Coalgebra numbers

CHAPTER 4

Game comonads and logical equivalences

This chapter is yet to be written.

4.1. Paths and embeddings

4.2. Bisimilarity

4.3. Equivalence in the full fragments

4.4. The equality symbol

4.5. Open pathwise embeddings

APPENDIX A

Category theory

A.1. Categories

Definition A.1. A *category* consists of:

- A class $\mathrm{Ob}(\mathcal{C})$ of *objects*, typically denoted by A, B, C.
- A class $\mathrm{Mor}(\mathcal{C})$ of *morphisms* (also called *arrows*), typically denoted by f, g, h.
- Two functions

$$\mathrm{dom}, \mathrm{cod}\colon \mathrm{Mor}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{C})$$

  assigning to a morphism its *domain* and *codomain*, respectively. We write $f\colon A \to B$ to indicate that f is a morphism with domain A and codomain B. For each pair of objects A, B, we define the associated *hom-set* to be the collection

$$\mathcal{C}(A, B) \coloneqq \{f \in \mathrm{Mor}(\mathcal{C}) \mid f\colon A \to B\}.$$

- For any triple of objects A, B, C, a *composition map*

$$\mathcal{C}(A, B) \times \mathcal{C}(B, C) \to \mathcal{C}(A, C), \quad (f, g) \mapsto g \circ f.$$

- For each object A, an *identity* morphism $\mathrm{id}_A\colon A \to A$.

The data above must satisfy the following equations for all arrows f, g, h and all objects A, B, whenever the compositions are well-defined:

$$h \circ (g \circ f) = (h \circ g) \circ f \qquad \text{(Associativity law)}$$

$$f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f \qquad \text{(Identity laws)}$$

Remark A.2. Properly speaking, the hom-sets $\mathcal{C}(A, B)$ need not be sets and may be *proper classes*. Cf. Definition A.25. ▲

Here is our first example of category:

- **Set**: the objects are sets and the morphisms are the functions between them.

We can always produce a new category from an old one by selecting some objects and taking all morphisms between them. For instance, we can define the category

- $\mathbf{Set}_{fin}$: the objects are *finite* sets and the morphisms are the functions between them.

In general, new categories can be obtained from old ones by restricting either their arrows or objects, or both.

Definition A.3. Suppose $\mathcal{C}$ is a category and consider collections $\mathrm{Ob}(\mathcal{D}) \subseteq \mathrm{Ob}(\mathcal{C})$ and, for all $A, B \in \mathrm{Ob}(\mathcal{D})$, $\mathcal{D}(A,B) \subseteq \mathcal{C}(A,B)$. Then $\mathcal{D}$ is a *subcategory* of $\mathcal{C}$ if

$$\mathrm{id}_A \in \mathcal{D}(A,A)$$

for all $A \in \mathrm{Ob}(\mathcal{D})$ and, for all $f \in \mathcal{D}(A,B)$ and $g \in \mathcal{D}(B,C)$,

$$g \circ f \in \mathcal{D}(A,C).$$

If, moreover, $\mathcal{D}(A,B) = \mathcal{C}(A,B)$ for all $A, B \in \mathrm{Ob}(\mathcal{D})$, then $\mathcal{D}$ is a *full subcategory* of $\mathcal{C}$.

For example, $\mathbf{Set}_{fin}$ is a full subcategory of $\mathbf{Set}$.

Exercise A.4. Show that a subcategory $\mathcal{D}$ of $\mathcal{C}$ is itself a category, with respect to the obvious composition maps and identity morphisms. ▲

In order to provide further examples of categories, we shall recall some basic mathematical notions.

σ-Structures. Recall that a *relational signature* σ is a set of relation symbols $\{R_i \mid i \in I\}$ such that each R_i is assigned a positive integer $\mathtt{ar}(R_i)$, called the *arity* of R_i. A *σ-structure* is given by a set A (the *universe* of the structure) together with an interpretation of the relation symbols in σ. That is, for each $R_i \in \sigma$ of arity n we have a set of n-tuples

$$R_i^A \subseteq A^n.$$

A *homomorphisms of σ-structures* (or *σ-homomorphism*, for short) from a σ-structure A to a σ-structure B is a function $f\colon A \to B$ between their universes that preserves the interpretations of the relations, i.e. for each $R_i \in \sigma$ of arity n and all $(a_1, \ldots, a_n) \in A^n$,

$$(a_1, \ldots, a_n) \in R_i^A \implies (f(a_1), \ldots, f(a_n)) \in R_i^B.$$

For any relational signature σ, we have a category

- $\mathbf{Str}(\sigma)$: the objects are σ-structures and the morphisms are the σ-homomorphisms.

Restricting to the *finite* σ-structures, i.e. those σ-structures whose universe is a finite set, we obtain the category

- $\mathbf{Str}_{fin}(\sigma)$: the objects are finite σ-structures and the morphisms are the σ-homomorphisms.

Relations. Let X, Y be any two sets. A *relation* from X to Y, written $X \nrightarrow Y$, is a subset R of the Cartesian product $X \times Y$. If $X = Y$, we simply refer to R as a (binary) relation on X. Given a pair $(x, y) \in X \times Y$, we sometimes write xRy instead of $(x, y) \in R$. The *identity relation* on a set X is the *diagonal relation*

$$\Delta_X \coloneqq \{(x, x) \in X \times X \mid x \in X\},$$

and the composition of two relations $R\colon X \nrightarrow Y$ and $S\colon Y \nrightarrow Z$ is the relation $R;S\colon X \nrightarrow Z$ defined by

$$R;S \coloneqq \{(x, z) \in X \times Z \mid \exists y \in Y.\ xRy \text{ and } ySz\}.$$

These data define a category

- **Rel**: the objects are sets and the morphisms are relations.

Set is a subcategory of **Rel**, but not a full subcategory.

Partial orders. A *partial order* on a set X is a binary relation on X satisfying the following conditions for all $x, y, z \in X$:

$$x \leq x \tag{Reflexivity}$$

$$x \leq y \ \wedge \ y \leq z \implies x \leq z \tag{Transitivity}$$

$$x \leq y \ \wedge \ y \leq x \implies x = y. \tag{Antisymmetry}$$

A *partially ordered set* (or *poset*, for short) is a pair $(X, \leq)$ where X is a set and $\leq$ is a partial order on X. A *monotone* (or *order-preserving*) map from a poset $(X, \leq_X)$ to a poset $(Y, \leq_Y)$ is a function $f\colon X \to Y$ such that $x_1 \leq_X x_2$ implies $f(x_1) \leq_Y f(x_2)$ for all $x_1, x_2 \in X$.

We obtain a category

- **Pos**: the objects are posets and the morphisms are monotone maps.

More generally, a relation that is reflexive and transitive—but not necessarily antisymmetric—is a called a *preorder*. In the same spirit as above, we can define a *preordered set* as a pair $(X, \leq)$ consisting of a set X and a preorder $\leq$ on it. This yields a category

- **PreOrd**: the objects are preordered sets and the morphisms are monotone maps.

Pos is a full subcategory of **PreOrd**.

Forests and trees. Let $(X, \leq)$ be a poset. A subset $C \subseteq X$ is said to be a *linear order* (or a *chain*) if, for all $x, y \in C$, either $x \leq y$ or $y \leq x$. For any element $x \in X$, let us write

$$\downarrow x \coloneqq \{y \in X \mid y \leq x\}$$

for the *downset* of x.

A *forest* is a poset $(X, \leq)$ such that, for all $x \in X$, the downset $\downarrow x$ is a finite linear order. The *roots* of a forest are the minimal elements, i.e. those elements that are not strictly above any other element. The *covering*

relation $\prec$ associated with the partial order $\leq$ is defined by $x \prec y$ if and only if $x < y$ (that is, $x \leq y$ and $x \neq y$) and there is no z such that $x < z < y$.

A *forest morphism* from a forest $(X, \leq_X)$ to a forest $(Y, \leq_Y)$ is a function $f\colon X \to Y$ that preserves roots and the covering relation. That is, for all $x_1, x_2 \in X$, if x_1 is a root of $(X, \leq_X)$ then $f(x_1)$ is a root of $(Y, \leq_Y)$, and if $x_1 \prec x_2$ then $f(x_1) \prec f(x_2)$. This defines a category

- **Forests**: the objects are forests and the arrows are forest morphisms.

A forest with at most one root is a called a *tree*. Considering only those objects of **Forests** that are (non-empty) trees, we obtain a new category

- **Trees**: the objects are *non-empty* trees and the arrows are forest morphisms.

The category of trees is a full subcategory of the category of forests.

Monoids and groups. A *monoid* is a triple $(M, \cdot, 1)$ where M is a set,

$$\_ \cdot \_ \colon M \times M \to M$$

is a binary operation on M (the *multiplication* of M), and 1 is an element of M (the *identity element*), satisfying the equational axioms:

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \qquad \text{(Associativity law)}$$
$$x \cdot 1 = x = 1 \cdot x \qquad \text{(Identity laws)}$$

A *monoid homomorphism* from $(M, \cdot_M, 1_M)$ to $(N, \cdot_N, 1_N)$ is a function $f\colon M \to N$ that preserves the multiplication and the identity element, i.e. for all $m_1, m_2 \in M$

$$f(m_1 \cdot_M m_2) = f(m_1) \cdot_N f(m_2) \quad \text{and} \quad f(1_M) = 1_N.$$

We thus get a category

- **Mon**: the objects are monoids and the arrows are monoid homomorphisms.

Moreover, a *group* is a monoid $(G, \cdot, 1)$ in which every element has an inverse. That is, for all $g \in G$ there is an element $g^{-1} \in G$ such that

$$g \cdot g^{-1} = 1 = g^{-1} \cdot g.$$

A *group homomorphism* from a group $(G, \cdot_G, 1_G)$ to a group $(H, \cdot_H, 1_H)$ is a function $f\colon G \to H$ that preserves the multiplication and the identity element (i.e., is a monoid homomorphisms), as well as inverses: for all $g \in G$, $f(g)^{-1} = f(g^{-1})$. This yields a category

- **Grp**: the objects are groups and the arrows are group homomorphisms.

Exercise A.5. Prove that **Grp** is a full subcategory of **Mon**. ▲

Elementary classes. Let T be a first-order theory in a signature τ (possibly containing both relation and function symbols). A *τ-homomorphism* between τ-structures is a function that preserves all relation and function symbols in τ. These data define a category

- $\mathbf{Mod}(T)$: the objects are models of T and the arrows are the τ-homomorphisms between them.

Note that the categories **Pos**, **PreOrd** and **Mon** are of the form $\mathbf{Mod}(T)$ for an appropriate signature τ and theory T. The same holds for **Grp**, because every monoid homomorphism between two groups is a group homomorphism (see Exercise A.5). On the other hand, **Rel**, **Forests** and **Trees** are not of the form $\mathbf{Mod}(T)$, for a signature consisting of a single binary relation symbol, since the arrows in these categories are not all "structure-preserving functions" between the appropriate objects.

Finally, every poset can be regarded as a category; this simple observation leads to a considerable source of examples for many categorical notions:

Example A.6. Any partially ordered set $(P, \leq)$ can be seen as a category in the following way: the objects are the elements of P and, for all $x, y \in P$, the hom-set $P(x, y)$ is given by

$$P(x,y) \coloneqq \begin{cases} \{\star\} & \text{if } x \leq y \\ \emptyset & \text{otherwise.} \end{cases}$$

In particular, between any two objects there is at most one arrow.[1] The reflexivity law $x \leq x$ yields the identity morphisms, and the transitivity law $x \leq y \leq z \Rightarrow x \leq z$ gives compositions of morphisms.

Note that we did not use the fact that posets satisfy the antisymmetry law, hence this construction works more generally for preordered sets. ▲

A.2. Reasoning with arrows

In ordinary mathematics, based on set-theoretic foundations, we are used to element-wise reasoning. In order to show that two functions $f, g\colon X \to Y$ are distinct, we seek to find an element $x \in X$ such that $f(x) \neq g(x)$. This hinges on the observation that two functions (with the same domain and codomain) are equal if, and only if, they coincide at each element of their domain. Similarly for continuous maps between topological spaces, group homomorphisms, linear maps between vector spaces, and so forth.

This simple principle, which is an intrinsic part of the "logic of sets" (and is a consequence of the *axiom of extensionality* in set theory), does not hold in all categories. In fact, the notion of *element* is not even available in

[1] This implies that *any* diagram in this category commutes!

an arbitrary category. For this reason, it is important to learn to reason in terms of arrows, rather than elements.[2]

A first useful observation is that, when comparing arrows, equations can be rephrased in terms of commutative diagrams. Consider for instance arrows $f\colon A \to B$, $g\colon B \to D$, $h\colon A \to C$ and $i\colon C \to D$. Then the equation

$$g \circ f = i \circ h$$

holds if, and only if, the following diagram commutes:

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle g} \\ C & \xrightarrow{i} & D \end{array}$$

This sort of rephrasing is at the base of *diagrammatic reasoning*, which is pervasive in category theory.

Many mathematical notions that are usually defined in terms of elements admit purely arrow-theoretic reformulations which can be used to generalise these concepts to arbitrary categories. Take, for instance, the notion of *bijection*: a function $f\colon X \to Y$ between sets is bijective if

$$\forall y \in Y\ \exists! x \in X.\ f(x) = y.$$

We can avoid any reference to the elements of X and Y by noting that f is a bijection precisely when there exists a function $g\colon Y \to X$ such that

$$g \circ f = \mathrm{id}_X \quad \text{and} \quad f \circ g = \mathrm{id}_Y.$$

The latter property only mentions arrows and makes sense in any category.

Definition A.7. An arrow $f\colon A \to B$ in a category $\mathcal{C}$ is an *isomorphism* if there exists an arrow $g\colon B \to A$ such that

$$g \circ f = \mathrm{id}_A \quad \text{and} \quad f \circ g = \mathrm{id}_B.$$

If there exists an isomorphism $A \to B$, we say that A is *isomorphic* to B and write $A \cong B$.

Exercise A.8. Show that the isomorphisms in the category **Mon** of monoids are precisely the bijective monoid homomorphisms. ▲

Exercise A.9. Prove that the isomorphisms in the category $\mathbf{Str}(\sigma)$, for any relational signature σ, are precisely the *σ-isomorphisms*, i.e. the bijective functions $f\colon A \to B$ such that, for each relation symbol $R \in \sigma$ of arity n,

$$(a_1, \ldots, a_n) \in R^A \iff (f(a_1), \ldots, f(a_n)) \in R^B.$$

[2]Having said that, *(generalised) elements* can be defined in a large class of categories, and the question of whether a category has *enough elements* is of interest in several contexts, such as topos theory or categorical logic. For example, Gödel's Completeness Theorem is equivalent to the statement that a certain category has enough elements.

Give an example of a bijective σ-homomorphism that is not an isomorphism. ▲

Exercise A.10. Describe the isomorphisms in the category **Pos** of posets. ▲

Note that every isomorphism f in a category satisfies the conditions

$$f \circ g = f \circ h \implies g = h$$

and

$$g \circ f = h \circ f \implies g = h$$

whenever the compositions are well defined. These cancellation laws define two important classes of morphisms in any category:

Definition A.11. Let $f\colon A \to B$ be an arrow in a category $\mathcal{C}$. We say that

- f is *monic* (or a *monomorphism*) if, for all arrows $g, h\colon C \to A$,

$$f \circ g = f \circ h \implies g = h.$$

- f is *epic* (or an *epimorphism*) if, for all arrows $g, h\colon B \to C$,

$$g \circ f = h \circ f \implies g = h.$$

Exercise A.12. Prove that, in **Set**, the monomorphisms and epimorphisms coincide, respectively, with the injective and surjective functions. Conclude that, in **Set**, a morphism that is both monic and epic is an isomorphism. ▲

Exercise A.13. Give an example of a category admitting an arrow that is both epic and monic, but not an isomorphism. ▲

Exercise A.14. Show that a morphism in **Mon** is monic if, and only if, it is an injective monoid homomorphism. Is every epimorphism in **Mon** surjective? ▲

Exercise A.15. Prove that, in the categories **Pos**, **Forests** and $\mathbf{Str}(\sigma)$, the monomorphisms and epimorphisms are those morphisms whose underlying function is, respectively, injective and surjective. ▲

By definition, a morphism in a category has a domain and a codomain; this is akin to the case of directed graphs, where edges have a source and a target. Given an arbitrary category $\mathcal{C}$, we can construct a new category $\mathcal{C}^{\mathrm{op}}$ by *reversing* the direction of arrows in $\mathcal{C}$. That is, an arrow $A \to B$ in $\mathcal{C}^{\mathrm{op}}$ is defined to be an arrow $B \to A$ in $\mathcal{C}$. Formally:

Definition A.16. The *opposite category* $\mathcal{C}^{\mathrm{op}}$ of a category $\mathcal{C}$ is defined by $\mathrm{Ob}(\mathcal{C}^{\mathrm{op}}) \coloneqq \mathrm{Ob}(\mathcal{C})$ and, for all objects A, B of $\mathcal{C}^{\mathrm{op}}$, $\mathcal{C}^{\mathrm{op}}(A, B) \coloneqq \mathcal{C}(B, A)$. Identities in $\mathcal{C}^{\mathrm{op}}$ are the same as in $\mathcal{C}$, and the composition $g \circ f$ in $\mathcal{C}^{\mathrm{op}}$ is defined as $f \circ g$ in $\mathcal{C}$.

For example, let $(P, \leq)$ be a poset regarded as a category as explained in Example A.6. Its opposite category can be identified with the poset $(P, \leq^{\mathrm{op}})$ obtained by turning the order of P upside down. That is, for all $x, y \in P$,

$$x \leq^{\mathrm{op}} y \iff y \leq x.$$

The passage from a category to its opposite is a purely formal operation but is at the heart of deep connections between e.g. algebra and geometry (mathematics), syntax and semantics (logic), and observables and states (physics). This is the subject of *duality theory*.

For now, it suffices to mention the following fact, sometimes referred to as *principle of duality*: An arrow-theoretic statement φ holds in a category $\mathcal{C}$ precisely when the dual statement (obtained from φ by reversing the direction of arrows) holds in $\mathcal{C}^{\mathrm{op}}$.

Exercise A.17. Prove that an arrow in a category $\mathcal{C}$ is monic (respectively, epic) if, and only if, it is epic (respectively, monic) in $\mathcal{C}^{\mathrm{op}}$. ▲

A.3. Functors

At the heart of category theory is the idea that morphisms between objects are as important as the object themselves. So, having defined the notion of category, it is natural to ask what is a "morphism of categories".

Definition A.18. A *functor* $F \colon \mathcal{C} \to \mathcal{D}$ from a category $\mathcal{C}$ to a category $\mathcal{D}$ consists of:

- A map $\mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{D})$ that assigns an object FA of $\mathcal{D}$ to every object A of $\mathcal{C}$.
- For all $A, B \in \mathrm{Ob}(\mathcal{C})$, a map $\mathcal{C}(A, B) \to \mathcal{D}(FA, FB)$ that assigns an arrow $Ff \colon FA \to FB$ in $\mathcal{D}$ to every arrow $f \colon A \to B$ in $\mathcal{C}$, preserving compositions and identities:

$$F(g \circ f) = Fg \circ Ff \quad \text{and} \quad F(\mathrm{id}_A) = \mathrm{id}_{FA}.$$

Trivial examples of functors are obtained by considering a subcategory $\mathcal{C}$ of a category $\mathcal{D}$. Then there is a functor $\mathcal{C} \to \mathcal{D}$ that acts as the inclusion on both the objects and arrows.

What is a functor from a poset P to a poset Q (regarded as categories)? The object map is simply a function $F \colon P \to Q$. On arrows, whenever $x, y \in P$ satisfy $x \leq y$, we must assign an arrow $Ff \colon Fx \to Fy$ to the unique arrow $x \to y$. But if there exists an arrow $Fx \to Fy$, it is unique. So, this amounts to saying that F is monotone. Note that the requirement that compositions and identities be preserved is trivially satisfied, as any diagram in a poset commutes. Therefore, functors between posets are precisely monotone maps.

Further examples of functors are presented in the following exercises.

Exercise A.19. For any set X, denote its power-set by $\mathcal{P}X$. Furthermore, given a function $f \colon X \to Y$ between sets, let $\mathcal{P}f \colon \mathcal{P}X \to \mathcal{P}Y$ be the *direct image map* that sends a subset $S \subseteq X$ to $f[S] \coloneqq \{f(x) \in Y \mid x \in S\}$.

Show that these assignments yield a functor $\mathcal{P}\colon \mathbf{Set} \to \mathbf{Set}$, known as the *(covariant) power-set functor*. ▲

Exercise A.20. Given a set X, let X^* denote the monoid of all finite *lists* (i.e., sequences) of elements of X. The monoid operation of X^* is given by concatenation of lists, and the identity element is the empty list. If $f\colon X \to Y$ is a function between sets, then $f^*\colon X^* \to Y^*$ sends a list $[x_1, \ldots, x_n]$ to $[f(x_1), \ldots, f(x_n)]$. Show that this construction determines a functor $\mathbf{Set} \to \mathbf{Mon}$. ▲

Exercise A.21. For any set X, let FX be the set of all *non-empty* finite lists of elements of X. We equip FX with a partial order defined as follows: for all $s, t \in FX$, $s \leq t$ if and only if s is a prefix of t. Any function $f\colon X \to Y$ induces a map $FX \to FY$ that sends a (non-empty) sequence $[x_1, \ldots, x_n]$ to $[f(x_1), \ldots, f(x_n)]$. Prove that these data define a functor $\mathbf{Set} \to \mathbf{Forests}$. ▲

Exercise A.22. Given a forest X, let X_+ be the tree obtained by adding a least element to X. Any forest morphism $X \to Y$ can be extended to a forest morphism $X_+ \to Y_+$ by sending the root of X_+ to the root of Y_+. Check that this gives a functor $\mathbf{Forests} \to \mathbf{Trees}$. ▲

Many examples of functors arise from "forgetting" part of the structure; these are generally (and informally) referred as *forgetful functors*. For example, given a group $(G, \cdot_G, 1_G)$, we can forget its algebraic structure and only retain the information about its underlying set G. Similarly, if f is a group homomorphism from $(G, \cdot_G, 1_G)$ to $(H, \cdot_H, 1_H)$, we can simply regard f as a function $G \to H$. These assignments determine a forgetful functor

$$\mathbf{Grp} \to \mathbf{Set}.$$

Likewise, we can define a functor $\mathbf{Grp} \to \mathbf{Mon}$ by only retaining the monoid structure of groups and group homomorphisms.

In the same way that morphisms in a category can be composed (whenever the codomain of one matches the domain of the other), functors between categories can be composed. If $F\colon \mathcal{B} \to \mathcal{C}$ and $G\colon \mathcal{C} \to \mathcal{D}$ are functors, then there is a *composite functor*

$$GF\colon \mathcal{B} \to \mathcal{D}$$

that sends an object A of $\mathcal{B}$ to the object GFA of $\mathcal{D}$, and an arrow $f\colon A \to B$ in $\mathcal{B}$ to the arrow $GFf\colon GFA \to GFB$ in $\mathcal{D}$.

Notation A.23. Given a functor $F\colon \mathcal{C} \to \mathcal{C}$, we sometimes denote the composite $FF\colon \mathcal{C} \to \mathcal{C}$ by F^2. Similarly, F^3 stands for FFF, and so forth.

Moreover, any category $\mathcal{C}$ admits an *identity functor*

$$\mathrm{id}_{\mathcal{C}}\colon \mathcal{C} \to \mathcal{C}$$

that acts as the identity on both objects and morphisms. This suggests that categories, together with functors between them, form themselves a

category. This is indeed the case, but a precise definition requires extra care so as to avoid *Russell's paradox* ("the set of all sets is not a set").

Exercise A.24. Define a chain of functors

$$\mathbf{Forests} \to \mathbf{Pos} \to \mathbf{PreOrd} \to \mathbf{Set}$$

whose composition is the obvious forgetful functor $\mathbf{Forests} \to \mathbf{Set}$. ▲

An important role is played by *set-valued functors*, i.e. functors $\mathcal{C} \to \mathbf{Set}$. If we regard the category of sets and functions as the "universe" where ordinary (classical) mathematics is carried out, set-valued functors correspond to interpretations (e.g. of theories) in this universe. This is the perspective adopted in *categorical logic*, where models of a theory are defined as set-valued functors satisfying appropriate properties. It is therefore pertinent to ask if, for any category $\mathcal{C}$, there are any functors $\mathcal{C} \to \mathbf{Set}$. The answer is *yes*—whenever $\mathcal{C}$ satisfies a mild set-theoretic "smallness condition".

Definition A.25. Let $\mathcal{C}$ be a category and suppose that, for all $A, B \in \mathrm{Ob}(\mathcal{C})$, the collection $\mathcal{C}(A, B)$ is a set (as opposed to a proper class). Then $\mathcal{C}$ is said to be *locally small*.

Any object A of a locally small category $\mathcal{C}$ determines a *hom-set functor*

$$\mathcal{C}(A, -) \colon \mathcal{C} \to \mathbf{Set}$$

that sends an object B of $\mathcal{C}$ to the set $\mathcal{C}(A, B)$, and an arrow $f \colon B \to C$ in $\mathcal{C}$ to the function

$$\mathcal{C}(A, f) \colon \mathcal{C}(A, B) \to \mathcal{C}(A, C), \quad g \mapsto f \circ g.$$

Properties of functors. In the same way that we discussed properties of arrows (e.g., being monic, epic, or an isomorphism), it is useful to consider properties that a functor may, or may not, satisfy.

Definition A.26. A functor $F \colon \mathcal{C} \to \mathcal{D}$ is said to be *faithful* (respectively, *full*) if, for all objects A, B of $\mathcal{C}$, the map

$$\mathcal{C}(A, B) \to \mathcal{D}(FA, FB), \quad f \mapsto Ff$$

is injective (respectively, surjective).

Note that, if $\mathcal{C}$ is a subcategory of $\mathcal{D}$, then the inclusion functor $\mathcal{C} \to \mathcal{D}$ is always faithful, and is full precisely when $\mathcal{C}$ is a full subcategory of $\mathcal{D}$ (see Definition A.3).

Typically, forgetful functors are faithful but not full.

Exercise A.27. Prove that the forgetful functors $\mathbf{Mon} \to \mathbf{Set}$ and $\mathbf{Pos} \to \mathbf{Set}$ are faithful but not full. ▲

Exercise A.28. Consider the functors $\mathbf{Set} \to \mathbf{Mon}$ and $\mathbf{Set} \to \mathbf{Forests}$ introduced, respectively, in Exercise A.20 and Exercise A.21. Are they faithful? Are they full? ▲

Exercise A.29. Show that every functor preserves isomorphisms, but need not preserve monomorphisms nor epimorphisms. ▲

Note that the composition of faithful functors is again faithful, and the composition of full functors is full (check this!).

Definition A.30. A functor $F\colon \mathcal{C} \to \mathcal{D}$ is an *isomorphism* if there exists a functor $G\colon \mathcal{D} \to \mathcal{C}$ such that

$$GF = \mathrm{id}_{\mathcal{C}} \quad \text{and} \quad FG = \mathrm{id}_{\mathcal{D}}.$$

If it exists, the functor G in the previous definition is unique and referred to as the *inverse* of F.

Exercise A.31. Prove that the functor **Forests** $\to$ **Trees** defined in Exercise A.22 is an isomorphism. Describe its inverse. ▲

If a functor is an isomorphism, then it is full and faithful (why?). However, the notion of isomorphism is typically too strong. This leads to the weaker concept of *equivalence* which will be introduced in the next section (see Definition A.40).

A.4. Natural transformations

We have seen that categories consist of objects and morphisms between them, and moreover there is an appropriate notion of morphism between categories—namely, functors. One could go further and consider morphisms of morphisms of categories, morphisms between the latter, etc. This is the framework of *higher category theory*; in the present notes we shall only take one more step and discuss "morphisms between functors".

Definition A.32. A *natural tranformation* $\alpha\colon F \to G$ between functors $F, G\colon \mathcal{C} \to \mathcal{D}$ is a collection

$$\{\alpha_A\colon FA \to GA \mid A \in \mathrm{Ob}(\mathcal{C})\}$$

of arrows in $\mathcal{D}$ indexed by objects of $\mathcal{C}$ satisfying the following *naturality condition*: For all arrows $f\colon A \to B$ in $\mathcal{C}$, the following square commutes.

$$\begin{array}{ccc} FA & \xrightarrow{Ff} & FB \\ {\scriptstyle \alpha_A}\downarrow & & \downarrow{\scriptstyle \alpha_B} \\ GA & \xrightarrow{Gf} & GB \end{array}$$

The morphism α_A is called the *component of* α *at* A.

Example A.33. Consider the (covariant) power-set functor $\mathcal{P}\colon$ **Set** $\to$ **Set** defined in Exercise A.19, along with the identity functor $\mathrm{id}_{\mathbf{Set}}\colon$ **Set** $\to$ **Set**. We show that there is a natural transformation

$$\eta\colon \mathrm{id}_{\mathbf{Set}} \to \mathcal{P}$$

whose component at a set X is the function

$$\eta_X\colon X \to \mathcal{P}X, \quad x \mapsto \{x\}.$$

To this end, we must verify that the following square commutes for all functions $f\colon X \to Y$ between sets:

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle \eta_X}\downarrow & & \downarrow{\scriptstyle \eta_Y} \\ \mathcal{P}X & \xrightarrow{\mathcal{P}f} & \mathcal{P}Y \end{array}$$

In turn, this follows by observing that, for all $x \in X$,

$$(\eta_Y \circ f)(x) = \eta_Y(f(x)) = \{f(x)\} = \mathcal{P}f(\{x\}) = (\mathcal{P}f \circ \eta_X)(x). \quad \blacktriangle$$

Exercise A.34. As in the previous example, consider the (covariant) power-set functor $\mathcal{P}\colon \mathbf{Set} \to \mathbf{Set}$. Show that there exists a natural transformation

$$\mu\colon \mathcal{P}\mathcal{P} \to \mathcal{P}$$

whose component at a set X is the function

$$\mu_X\colon \mathcal{P}\mathcal{P}X \to \mathcal{P}X, \quad S \mapsto \bigcup S. \quad \blacktriangle$$

Exercise A.35. Let $F\colon \mathbf{Set} \to \mathbf{Mon}$ be the functor defined in Exercise A.20, and let $U\colon \mathbf{Mon} \to \mathbf{Set}$ be the forgetful functor. Recall that, for all sets X, FX is the monoid of finite lists of elements of X, hence UFX is the *set* of finite lists of elements of X. Denoting the composite functor by $T \coloneqq UF\colon \mathbf{Set} \to \mathbf{Set}$, check that there is a natural transformation

$$\eta\colon \mathrm{id}_{\mathbf{Set}} \to T$$

whose component at a set X is the function

$$\eta_X\colon X \to TX, \quad x \mapsto [x].$$

Moreover, show that there is a natural transformation

$$\mu\colon TT \to T$$

whose component at X is the *flatten map*

$$\mu_X\colon TTX \to TX,$$
$$[[x_{1,1}, \ldots, x_{1,n_1}], \ldots, [x_{k,1}, \ldots, x_{k,n_k}]] \mapsto [x_{1,1}, \ldots, x_{1,n_1}, \ldots, x_{k,1}, \ldots, x_{k,n_k}].$$

Note the similarity between these natural transformations and those defined for the power-set functor $\mathcal{P}$ in Example A.33 and Exercise A.34. These are two instances of the same concept, namely that of *monad*. See Appendix A.5. $\blacktriangle$

Exercise A.36. Let $F\colon \mathbf{Set} \to \mathbf{Forests}$ be the functor defined in Exercise A.21, and consider the composite functor $G \coloneqq UF\colon \mathbf{Set} \to \mathbf{Set}$ where $U\colon \mathbf{Forests} \to \mathbf{Set}$ is the forgetful functor. For every set X, GX is the set of all non-empty finite lists of elements of X. Verify that there is a natural transformation

$$\varepsilon\colon G \to \mathrm{id}_{\mathbf{Set}}$$

whose component at X is the map sending a list to its last element:

$$\varepsilon_X \colon GX \to X, \quad [x_1, \ldots, x_n] \to x_n.$$

(Note that ε_X is well defined because FX consists of *non-empty* lists!)

Furthermore, show that there is a natural transformation

$$\delta \colon F \to GG$$

whose component at X sends a list to the list of its (non-empty) prefixes:

$$\delta_X \colon GX \to GGX, \quad [x_1, \ldots, x_n] \mapsto [[x_1], [x_1, x_2], \ldots, [x_1, \ldots, x_n]]. \quad \blacktriangle$$

Functor categories. To give a precise meaning to the assertion that natural transformations are morphisms of functors, we introduce the notion of *functor category*. For any two categories $\mathcal{C}, \mathcal{D}$, there is a category

$$[\mathcal{C}, \mathcal{D}]$$

whose objects are functors $F \colon \mathcal{C} \to \mathcal{D}$ and whose arrows are natural transformations. Given a functor $F \colon \mathcal{C} \to \mathcal{D}$, its identity is the natural transformation $\alpha \colon F \to F$ such that, for all objects A of $\mathcal{C}$, $\alpha_A = \mathrm{id}_{FA}$. Natural transformations can be composed in the obvious way: if $\alpha \colon F \to G$ and $\beta \colon G \to H$ are natural transformations, then their composite is the natural transformation

$$\beta\alpha \colon F \to H, \quad (\beta \circ \alpha)_A \coloneqq \beta_A \circ \alpha_A.$$

Exercise A.37. Verify the statements in the previous paragraph and check carefully that $[\mathcal{C}, \mathcal{D}]$ is indeed a category. ▲

What are the isomorphisms in a functor category?

Definition A.38. Let $\alpha \colon F \to G$ be a natural transformation between functors $F, G \colon \mathcal{C} \to \mathcal{D}$. If all components of α are isomorphisms in $\mathcal{D}$, then α is called a *natural isomorphism*.

Exercise A.39. Prove that the isomorphisms in a functor category $[\mathcal{C}, \mathcal{D}]$ are precisely the natural isomorphisms. ▲

We can use the notion of natural isomorphism to weaken the concept of isomorphism between categories. This is akin to the passage from homeomorphism to homotopy equivalence in topology.

Definition A.40. A functor $F \colon \mathcal{C} \to \mathcal{D}$ is an *equivalence* if there exists a functor $G \colon \mathcal{D} \to \mathcal{C}$ and natural isomorphisms

$$\alpha \colon GF \to \mathrm{id}_{\mathcal{C}} \quad \text{and} \quad \beta \colon FG \to \mathrm{id}_{\mathcal{D}}.$$

If there exists an equivalence $\mathcal{C} \to \mathcal{D}$, we shall say that $\mathcal{C}$ is *equivalent* to $\mathcal{D}$ and write $\mathcal{C} \simeq \mathcal{D}$.

When $\mathcal{D} = \mathbf{Set}$, the objects of $[\mathcal{C}, \mathbf{Set}]$ are set-valued functors defined on $\mathcal{C}$. Likewise, one can consider the functor category $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$ of set-valued functors defined on the opposite category $\mathcal{C}^{\mathrm{op}}$. These play a central role in category theory and are called *presheaves on* $\mathcal{C}$. The category of presheaves

on $\mathcal{C}$ is in general much larger than the original category $\mathcal{C}$, yet it admits a copy of $\mathcal{C}$ as a full subcategory. This is known as the *Yoneda embedding*, which we recall below.

Exercise A.41. Let **2** denote a category with precisely two objects, their identities, and two distinct parallel morphisms. This category can be depicted as follows (where we omit the identity arrows for convenience):

$$\bullet \rightrightarrows \bullet$$

Show that the functor category $[\mathbf{2}, \mathbf{Set}]$ is isomorphic to the category of simple, directed (multi)graphs and graph homomorphisms. ▲

Exercise A.42. Consider the ordered set $(\mathbb{N}, \leq)$ of natural numbers as a category (see Example A.6). Prove that the category $[\mathbb{N}^{\mathrm{op}}, \mathbf{Set}]$ of presheaves on $\mathbb{N}$ is equivalent to **Forests**. ▲

Recall the notion of hom-set functor defined on Page 30. Every object of a locally small category $\mathcal{C}$ induces a presheaf

$$\mathcal{C}^{\mathrm{op}}(A, -) = \mathcal{C}(-, A) \colon \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}.$$

Further, any arrow $f \colon A \to B$ in $\mathcal{C}$ induces a natural transformation

$$\mathcal{C}(-, A) \to \mathcal{C}(-, B)$$

whose component at an object C of $\mathcal{C}$ (equivalently, of $\mathcal{C}^{\mathrm{op}}$) is the function

$$\mathcal{C}(C, A) \to \mathcal{C}(C, B), \quad g \mapsto f \circ g.$$

(Check that this is indeed a natural transformation!) This determines a functor $\mathcal{C} \to [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$.

THEOREM A.43 (Yoneda embedding). *Let $\mathcal{C}$ be a locally small category. The functor*

$$\mathcal{C} \to [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}], \quad A \mapsto \mathcal{C}(-, A) \colon \mathcal{C}^{\mathrm{op}} \to \mathbf{Set}$$

is full and faithful.

A.5. Monads and comonads

Together with the notions of *functor* and *natural transformation* (and that of *adjunction*, to be discussed in Appendix A.7), the concept of *monad* is one of the pillars of basic category theory. Its importance is in large part due to the fact that it shows how a number of constructions throughout mathematics are instances of the same abstract notion. Monads are pervasive in algebra, but appear also in topology, probability theory, functional programming, and other areas. The dual notion, that of *comonad*, is equally important but perhaps less familiar to many researchers. We shall present both notions, but focus in particular on comonads as they play a key role in relation with finite model theory.

Monads and comonads are functors of type $\mathcal{C} \to \mathcal{C}$, i.e. from a category to itself, satisfying appropriate properties. The intuitions are very different

though: whereas monads encode ways to *combine* distinct parts or elements, comonads correspond to *decompositions* (or *unravellings*) of an object.

To make this idea more precise, let us look at the case of monads first. Consider a monoid $(M, \cdot, 1)$: the monoid operation $\cdot$ tells us how to combine any two elements of M, and 1 is the identity element for this operation. In a sense, monads are generalised monoids.[3] As such, they come equipped with a multiplication and an identity satisfying the usual monoid laws.

Definition A.44. A *monad* on a category $\mathcal{C}$ is a tuple (T, μ, η) where

- $T\colon \mathcal{C} \to \mathcal{C}$ is a functor,
- $\mu\colon T^2 \to T$ is a natural transformation, called *multiplication*,
- $\eta\colon \mathrm{id}_{\mathcal{C}} \to T$ is a natural transformation, called *unit*,

such that the following diagrams commute for all objects A of $\mathcal{C}$:

$$\begin{array}{ccc} T^3A & \xrightarrow{\mu_{TA}} & T^2A \\ {\scriptstyle T\mu_A}\downarrow & & \downarrow{\scriptstyle \mu_A} \\ T^2A & \xrightarrow{\mu_A} & TA \end{array} \qquad \begin{array}{ccc} TA & \xrightarrow{\eta_{TA}} & T^2A \\ {\scriptstyle T\eta_A}\downarrow & \searrow^{\mathrm{id}_{TA}} & \downarrow{\scriptstyle \mu_A} \\ T^2A & \xrightarrow{\mu_A} & TA \end{array}$$

In the previous definition, T^2 denotes the composite functor TT, and similarly for T^3; see Notation A.23.

Example A.45. Recall from Exercise A.35 that there is a functor $T\colon \mathbf{Set} \to \mathbf{Set}$ that assigns to a set X the set of all finite lists of elements of X, and it comes equipped with natural transformations

$$\mu\colon T^2 \to T \quad \text{and} \quad \eta\colon \mathrm{id}_{\mathbf{Set}} \to T.$$

The components of μ are the flatten maps that transform a list of lists into a list, e.g. $[[x, y], [z], [y]]$ is sent to $[x, y, z, y]$, and the components of η send an element x to the one-element list $[x]$.

The tuple (T, μ, η) is a monad on $\mathbf{Set}$, called the *free monoid monad*. This amounts to saying that the diagrams in Definition A.49 commute for all objects of $\mathbf{Set}$. We give a "proof by example"; the formal proof follows the same ideas (with some extra bookkeeping) and is left to the reader. Suppose we have a set $X = \{x, y, z\}$. For the left-hand diagram in Definition A.49, we must consider an element of T^3X, i.e. a list of lists of lists of elements of X. For instance,

$$[[[x], [x, y]], [[x, y, z]]].$$

Chasing this element around the diagram, we get:

$$\begin{array}{ccc} [[[x], [x, y]], [[x, y, z]]] & \xmapsto{\mu_{TX}} & [[x], [x, y], [x, y, z]] \\ {\scriptstyle T\mu_X}\downarrow & & \downarrow{\scriptstyle \mu_X} \\ [[x, x, y], [x, y, z]] & \xmapsto{\mu_X} & [x, x, y, x, y, z] \end{array}$$

[3] This is a small lie: the truth is that a monad *is* (a special case of) a monoid, namely a monoid object in the functor category $[\mathcal{C}, \mathcal{C}]$.

The right-hand diagram in Definition A.49 is easier to check: given an element of TX, say $[z, y, y, x]$, we have:

$$\begin{array}{ccc} [z,y,y,x] & \xmapsto{\eta_{TX}} & [[z,y,y,x]] \\ {\scriptstyle T\eta_X}\downarrow & \searrow^{\mathrm{id}_{TX}} & \downarrow{\scriptstyle \mu_X} \\ [[z],[y],[y],[x]] & \xmapsto{\mu_X} & [z,y,y,x] \end{array}$$

▲

Exercise A.46. Fill in the details of the proof in Example A.45. ▲

It is instructive to look at what is a monad on a poset (regarded as a category according to Example A.6). Let $(P, \leq)$ be a poset, and let (T, μ, η) be a monad on P. Then $T\colon P \to P$ is a monotone map (cf. the discussion on Page 28), and for each $x \in P$ the component of μ at x yields an arrow $\mu_x\colon T^2x \to Tx$. That is,

$$T^2x \leq Tx.$$

Similarly, the component of η at x yields an arrow $\eta_x\colon x \to Tx$ and so

$$x \leq Tx.$$

By monotonicity, applying T to both sides of the latter inequation we obtain $Tx \leq T^2x$ and therefore

$$T^2x = Tx.$$

In other words, T is a *closure operator* on P.

Definition A.47. A *closure operator* on a poset P is a monotone map $t\colon P \to P$ that is

(1) *increasing*, i.e. for all $x \in P$, $x \leq tx$, and
(2) *idempotent*, i.e. for all $x \in P$, $t^2x = tx$.

Exercise A.48. Show that monads on posets are precisely the closure operators. That is, for any closure operator $t\colon P \to P$ there exist unique natural transformations μ and η such that (t, μ, η) is a monad on P. ▲

Therefore, a monad on a poset is akin to the modality $\Diamond$ in modal logic, and dually the modal operator $\Box$ is an instance of a comonad.

A comonad on a category $\mathcal{C}$ can be succinctly defined as a monad on the opposite category $\mathcal{C}^{\mathrm{op}}$. More explicitly:

Definition A.49. A *comonad* on a category $\mathcal{C}$ is a tuple (G, δ, ε) where

- $G\colon \mathcal{C} \to \mathcal{C}$ is a functor,
- $\delta\colon G \to G^2$ is a natural transformation, called *comultiplication*,
- $\varepsilon\colon G \to \mathrm{id}_{\mathcal{C}}$ is a natural transformation, called *counit*,

such that the following diagrams commute for all objects A of $\mathcal{C}$:

$$\begin{array}{ccc} GA & \xrightarrow{\delta_A} & G^2A \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_{GA}} \\ G^2A & \xrightarrow{G\delta_A} & G^3A \end{array} \qquad \begin{array}{ccc} GA & \xrightarrow{\delta_A} & G^2A \\ {\scriptstyle \delta_A}\downarrow & \searrow^{\mathrm{id}_{GA}} & \downarrow{\scriptstyle \varepsilon_{GA}} \\ G^2A & \xrightarrow{G\varepsilon_A} & GA \end{array}$$

When defining comonads (and similarly, monads), there are a number of things to be verified: one should give a functor, two natural transformations, and check that the appropriate diagrams commute. We now recall an equivalent description of comonads that allows us to reduce these verifications and is very useful in concrete cases; a similar description for monads is of course available using the principle of duality.

Definition A.50. A *comonad in Kleisli–Manes form* on a category $\mathcal{C}$ is given by:

- an object map $G\colon \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{C})$,
- a morphism $\varepsilon_A\colon GA \to A$ for every $A \in \mathrm{Ob}(\mathcal{A})$,
- a *coextension operation* associating with any morphism $f\colon GA \to B$ a morphism $f^*\colon GA \to GB$.

These must satisfy the following equations for all morphisms $f\colon GA \to B$ and $g\colon GB \to C$:

$$\varepsilon_A^* = \mathrm{id}_{GA}, \quad \varepsilon_B \circ f^* = f, \quad (g \circ f^*)^* = g^* \circ f^*. \tag{A.1}$$

Given a comonad in Kleisli–Manes form, we can extend the object map $G\colon \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{C})$ to a functor $\mathcal{C} \to \mathcal{C}$ by setting

$$Gf \coloneqq (f \circ \varepsilon_A)^*$$

for every morphism $f\colon A \to B$. Furthermore, the arrows $\delta_A \coloneqq \mathrm{id}_{GA}^*$ are the components of a natural transformation $\delta\colon G \to G^2$, the arrows ε_A are the components of a natural transformation $\varepsilon\colon G \to \mathrm{id}_{\mathcal{C}}$, and (G, δ, ε) is a comonad on $\mathcal{C}$.

Exercise A.51. Prove that, conversely, every comonad on $\mathcal{C}$ induces a comonad in Kleisli–Manes form, and the two assignments are inverse to each other. (Hint: define the coextension of $f\colon GA \to B$ as $f^* \coloneqq Gf \circ \delta_A$.) ▲

Example A.52. We define a comonad on **Set** using the Kleisli–Manes form. The object map

$$G\colon \mathrm{Ob}(\mathbf{Set}) \to \mathrm{Ob}(\mathbf{Set})$$

sends a set X to the set of all non-empty finite lists of elements of X. For each set X,

$$\varepsilon_X\colon GX \to X, \quad [x_1, \ldots, x_n] \mapsto x_n$$

is the function sending a list to its last element. Finally, the coextension operation sends a function $f\colon GX \to Y$ to the function

$$f^*\colon GX \to GY, \quad [x_1, \ldots, x_n] \mapsto [f([x_1]), f([x_1, x_2]), \ldots, f([x_1, \ldots, x_n])].$$

It remains to show that the equations

$$\varepsilon^*_X = \mathrm{id}_{GX}, \quad \varepsilon_Y \circ f^* = f, \quad (g \circ f^*)^* = g^* \circ f^*$$

are satisfied for all functions $f\colon GX \to Y$ and $g\colon GY \to Z$. So, we fix an arbitrary element $[x_1, \ldots, x_n] \in GX$ and compute:

$$\begin{aligned}\varepsilon^*_X([x_1, \ldots, x_n]) &= [\varepsilon_X([x_1]), \varepsilon_X([x_1, x_2]), \ldots, \varepsilon_X([x_1, \ldots, x_n])] \\ &= [x_1, \ldots, x_n],\end{aligned}$$

$$\begin{aligned}(\varepsilon_Y \circ f^*)([x_1, \ldots, x_n]) &= \varepsilon_Y([f([x_1]), \ldots, f([x_1, \ldots, x_n])]) \\ &= f([x_1, \ldots, x_n]),\end{aligned}$$

$$\begin{aligned}(g \circ f^*)^*([x_1, \ldots, x_n]) &= [(g \circ f^*)([x_1]), \ldots, (g \circ f^*)([x_1, \ldots, x_n])] \\ &= [g([f(x_1)]), \ldots, g([f([x_1]), \ldots, f([x_1, \ldots, x_n])])] \\ &= g^*([f([x_1]), \ldots, f([x_1, \ldots, x_n])]) \\ &= (g^* \circ f^*)([x_1, \ldots, x_n]).\end{aligned}$$

The associated comonad (G, δ, ε) is the one described in Exercise A.36 (check this!). ▲

Exercise A.53. Prove that comonads on a poset P are precisely the *interior operators* on P, i.e. the monotone maps $g\colon P \to P$ that are

(1) *decreasing*, i.e. $gx \leq x$ for all $x \in P$, and
(2) *idempotent*, i.e. $g^2x = gx$ for all $x \in P$.

Either give a direct proof or use Exercise A.48 combined with the principle of duality. ▲

A.6. Kleisli and Eilenberg–Moore categories for a comonad

Monads and comonads induce, respectively, categories of *algebras* and categories of *coalgebras*. For example, algebras for monads over **Set** essentially correspond to varieties of algebras in the sense of universal algebra.[4] In this section, we shall focus exclusively on coalgebras for comonads, as these are relevant in connection with finite model theory.

Given a comonad, there are two categories of coalgebras that are worth looking at: the *Kleisli category* is the "minimal" one and consists only of the *co-free* coalgebras, whereas the *Eilenberg–Moore category* is the "maximal" one and consists of all coalgebras. The former is easier to describe, but some constructions require working in the latter category. Let us start by introducing the Kleisli category of a comonad:

Definition A.54. Let G be a comonad (in Kleisli–Manes form) on a category $\mathcal{C}$. The *Kleisli category* of G, denoted by $\mathbf{K}(G)$, is defined as follows:

- $\mathrm{Ob}(\mathbf{K}(G)) = \mathrm{Ob}(\mathcal{C})$.

[4] To make the statement precise, one should restrict to those monads on **Set** that are *finitary*, i.e. that preserve so-called *directed colimits*.

- For all $A, B \in \mathrm{Ob}(\mathcal{C})$, $\mathbf{K}(G)(A,B) = \mathcal{C}(GA, B)$.

For any two arrows $f \in \mathbf{K}(G)(A,B)$ and $g \in \mathbf{K}(G)(B,C)$, their composite is defined as the following composition in $\mathcal{C}$:

$$GA \xrightarrow{f^*} GB \xrightarrow{g} C.$$

The identity $\mathrm{id}_A \in \mathbf{K}(G)(A,A)$ is the arrow $\varepsilon_A \colon GA \to A$ in $\mathcal{C}$.

Exercise A.55. Verify that $\mathbf{K}(G)$ is a category. That is, the composition operation is associative and the identity arrows are identities for the composition operation. ▲

To get a better intuition of the Kleisli category, it is useful to compare it to the Eilenberg–Moore category.

Definition A.56. Le G be a comonad on a category $\mathcal{C}$. An *Eilenberg–Moore coalgebra* for G is a pair (A, α) such that $A \in \mathrm{Ob}(\mathcal{C})$, $\alpha \in \mathcal{C}(A, GA)$, and the following diagrams commute.

$$\begin{array}{ccc} A & \xrightarrow{\alpha} & GA \\ & \searrow^{\mathrm{id}_A} & \downarrow{\scriptstyle \varepsilon_A} \\ & & A \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{\alpha} & GA \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \delta_A} \\ GA & \xrightarrow{G\alpha} & G^2A \end{array}$$

The arrow α is called the *structure map* of the coalgebra. A *morphism of Eilenberg–Moore coalgebras* $(A, \alpha) \to (B, \beta)$ is an arrow $f \in \mathcal{C}(A,B)$ compatible with the structures maps, i.e. making the following square commute.

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \beta} \\ GA & \xrightarrow{Gf} & GB \end{array}$$

Definition A.57. Let G be a comonad on a category $\mathcal{C}$. The *Eilenberg–Moore category* of G, denoted by $\mathbf{EM}(G)$, consists of the Eilenberg–Moore coalgebras for G and their morphisms. Compositions and identities are the obvious ones.

For any comonad G, there is a functor

$$\nabla \colon \mathbf{K}(G) \to \mathbf{EM}(G).$$

At the level of objects, ∇ sends $A \in \mathrm{Ob}(\mathbf{K}(G)) = \mathrm{Ob}(\mathcal{C})$ to (GA, δ_A). With regards to morphisms, ∇ assigns to an arrow $f \in \mathcal{C}(GA, B)$ the arrow f^*.

Proposition A.58. *Let G be a comonad on a category $\mathcal{C}$. Then*

$$\nabla \colon \mathbf{K}(G) \to \mathbf{EM}(G)$$

is a full and faithful functor.

Proof. We first check that ∇ is well defined. Let A be an arbitrary object of $\mathbf{K}(G)$. To show that (GA, δ_A) is an Eilenberg–Moore coalgebra, we must prove that the following diagrams commute.

$$\begin{array}{ccc} GA & \xrightarrow{\delta_A} & G^2A \\ & \searrow^{\mathrm{id}_{GA}} & \downarrow{\scriptstyle \varepsilon_{GA}} \\ & & GA \end{array} \qquad \begin{array}{ccc} GA & \xrightarrow{\delta_A} & G^2A \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_{GA}} \\ G^2A & \xrightarrow{G\delta_A} & G^3A \end{array}$$

In turn, this follows at once from the fact that G is a comonad (see Definition A.49).

Now, fix an arbitrary arrow f in $\mathbf{K}(G)$, i.e. $f \in \mathcal{C}(GA, B)$. To see that $f^*\colon GA \to GB$ is a morphism of Eilenberg–Moore coalgebras, we must check that the following square commutes.

$$\begin{array}{ccc} GA & \xrightarrow{f^*} & GB \\ {\scriptstyle \delta_A}\downarrow & & \downarrow{\scriptstyle \delta_B} \\ G^2A & \xrightarrow{Gf^*} & G^2B \end{array}$$

To this end, recall that

$$f^* = Gf \circ \delta_A \tag{A.2}$$

for all arrows $f\colon GA \to B$ (see Exercise A.51). In particular, $\mathrm{id}^*_{GA} = \delta_A$. Thus,

$$\begin{aligned} Gf^* \circ \delta_A &= f^{**} && \text{Eq. (A.2)} \\ &= (\mathrm{id}_{GB} \circ f^*)^* \\ &= \mathrm{id}^*_{GB} \circ f^* && 3^{\text{rd}} \text{ equation in Eq. (A.1)} \\ &= \delta_B \circ f^*. \end{aligned}$$

The fact that ∇ preserves compositions and identities is an immediate consequence of the third and first equations, respectively, for a comonad in Kleisli–Manes form (check the details!).

It remains to show that ∇ is full and faithful. Faithfulness is clear: just observe that, for all $f, g \in \mathcal{C}(GA, B)$, $f^* = g^*$ implies

$$f = \varepsilon_B \circ f^* = \varepsilon_B \circ g^* = g$$

by virtue of the second equation for a comonad in Kleisli–Manes form. To establish fullness of ∇, let $g\colon (GA, \delta_A) \to (GB, \delta_B)$ be a morphism of Eilenberg–Moore coalgebras. We have

$$\begin{aligned} (\varepsilon_B \circ g)^* &= G(\varepsilon_B \circ g) \circ \delta_A && \text{Eq. (A.2)} \\ &= G\varepsilon_B \circ Gg \circ \delta_A && G \text{ is a functor} \\ &= G\varepsilon_B \circ \delta_B \circ g && g \text{ coalgebra morphism} \\ &= \varepsilon^*_B \circ g, && \text{Eq. (A.2)} \end{aligned}$$

which coincides with g by the first equation for comonads in Kleisli–Manes form. Note that $\varepsilon_B \circ g \in \mathcal{C}(GA, B) = \mathbf{K}(G)(A, B)$, hence ∇ is full. □

Let us look at an example. Consider the comonad G on **Set** defined in Example A.52. Recall that, for any set X, GX is the set of non-empty finite lists of elements of X, and a function $f\colon X \to Y$ is sent to the function

$$Gf\colon GX \to GY, \quad [x_1, \ldots, x_n] \mapsto [fx_1, \ldots, fx_n].$$

We claim that there is an isomorphism of categories

$$\mathbf{EM}(G) \cong \mathbf{Forests}.$$

Suppose (X, α) is an Eilenberg–Moore coalgebra for G. The set GX carries a natural forest order, namely the prefix order. The commutativity of the first diagram in Definition A.56 tells us that $\varepsilon_X \circ \alpha = \mathrm{id}_X$ and so the structure map $\alpha\colon X \to GX$ is injective. Hence the forest order on GX induces a partial order on X given by

$$x \leq y \iff \alpha(x) \leq \alpha(y)$$

for all $x, y \in X$. To show that this is a forest order on X, it suffices to show that the image of α is a *downwards closed* subset of GX (why?). That is, any element of GX that is below some element in the image of α is also in the image of α.

Fix an arbitrary $x \in X$ and suppose that $\alpha(x)$ is of the form $[x_1, \ldots, x_n]$ (incidentally, note that $x_n = x$ because $\varepsilon_X \circ \alpha = \mathrm{id}_X$). The commutativity of the second diagram in Definition A.56 implies that

$$[\alpha(x_1), \ldots, \alpha(x_n)] = [[x_1], \ldots, [x_1, \ldots, x_n]].$$

But any element that is below $\alpha(x)$ in the prefix order of GX is of the form $[x_1, \ldots, x_j]$ for some $j \in \{1, \ldots, n\}$, and in view of the previous equation all these elements are in the image of α.

Therefore, any structure map $\alpha\colon X \to GX$ defines a forest order on X. Further, the coalgebra morphisms preserve these forest orders. To see this, suppose that $f\colon (X, \alpha) \to (Y, \beta)$ is a morphism of coalgebras, i.e. the following square commutes.

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ GX & \xrightarrow{Gf} & GY \end{array}$$

An element $x \in X$ is a root precisely when $\alpha(x) = [x]$ (why?), and similarly for elements of Y. Thus, f preserves roots because $\alpha(x) = [x]$ entails

$$\beta(fx) = [fx].$$

To see that f preserves the covering relation, suppose that $x, x' \in X$ satisfy $x \prec x'$ and $\alpha(x') = [x_1, \ldots, x_n]$. Observe that $x \prec x'$ if and only if $fx \prec fx'$ (why?), and so $\alpha(x) = [x_1, \ldots, x_{n-1}]$. The commutativity of the previous square yields

$$\beta(fx) = [fx_1, \ldots, fx_{n-1}] \quad \text{and} \quad \beta(fx') = [fx_1, \ldots, fx_n]$$

which shows that $\beta(fx) \prec \beta(fx')$.

It is straightforward to check that compositions and identities are preserved, hence this construction of a forest order from a structure map gives a functor

$$\mathbf{EM}(G) \to \mathbf{Forests}.$$

Conversely, given a forest order $(X, \leq)$, let $\alpha\colon X \to GX$ be the function that sends $x \in X$ to the list $[x_1, \ldots, x_n]$ of (non-strict) predecessors of x. That is, x_1 is a root and

$$x_1 \prec \cdots \prec x_n = x.$$

The first diagram in Definition A.56 commutes simply because the last element of the list $\alpha(x)$ is x, and the second diagram commutes by definition of α (check this!). In other words, (X, α) is an Eilenberg–Moore coalgebra. Furthermore, any forest morphism $f\colon (X, \leq) \to (Y, \leq)$ preserves the corresponding structure maps because, for all $x \in X$, if the predecessors of x are $x_1 \prec \cdots \prec x$ then the predecessors of fx are $fx_1 \prec \cdots \prec fx$ (spell out the details of this argument!).

Again, it is clear that compositions and identities are preserved, so we obtain a functor

$$\mathbf{Forests} \to \mathbf{EM}(G).$$

Exercise A.59. Prove that the functors $\mathbf{EM}(G) \leftrightarrows \mathbf{Forests}$ defined above are inverse to each other. ▲

What is the Kleisli category $\mathbf{K}(G)$ of the comonad G? By Proposition A.58, combined with the previous discussion, it can be identified with the full subcategory of **Forests** defined by the forests of the form GX (endowed with the prefix order).

A.7. Adjunctions

Note. *This section will appear in due course. Familiarity with adjunctions is not necessary for most of the material in the course, and this notion will be recalled in the lectures when needed.*

Adjunctions are a fundamental notion in category theory and provide a vast generalisation of free constructions such as free groups, free modules, etc. This is related to the fact that every adjunction induces a monad—and every monad arises from an adjunction (not a unique one, though).

Importantly, adjunctions have a symmetric nature. In fact, every adjunction induces not only a monad but also a comonad, and every comonad arises from some adjunction.

Bibliography

Abramsky, S., Dawar, A., and Wang, P. (2017). The pebbling comonad in finite model theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS*, pages 1–12.

Abramsky, S. and Shah, N. (2018). Relating Structure and Power: Comonadic Semantics for Computational Resources. In *27th EACSL Annual Conference on Computer Science Logic*, volume 119 of *Leibniz International Proceedings in Informatics*, pages 2:1–2:17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Abramsky, S. and Shah, N. (2021). Relating structure and power: Comonadic semantics for computational resources. *Journal of Logic and Computation*, 31(6):1390–1428.

Abramsky, S. and Tzevelekos, N. (2011). Introduction to categories and categorical logic. In Coecke, B., editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 3–94. Springer-Verlag. Preprint available at https://arxiv.org/abs/1102.1313.

Adámek, J., Herrlich, H., and Strecker, G. E. (1990). *Abstract and concrete categories. The joy of cats*. Pure and Applied Mathematics (New York). John Wiley & Sons, Inc., New York. Online edition freely available at http://katmat.math.uni-bremen.de/acc/acc.pdf.

Barto, L. and Kozik, M. (2014). Constraint satisfaction problems solvable by local consistency methods. *J. ACM*, 61(1):3:1–3:19.

Barwise, J. (1977). On Moschovakis closure ordinals. *Journal of Symbolic Logic*, 42(2):292–296.

Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Casanovas, E., Dellunde, P., and Jansana, R. (1996). On elementary equivalence for equality-free logic. *Notre Dame J. Formal Logic*, 37(3):506–522.

Chandra, A. K. and Merlin, P. M. (1977). Optimal implementation of conjunctive queries in relational data bases. In Hopcroft, J. E., Friedman, E. P., and Harrison, M. A., editors, *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, pages 77–90. ACM.

Dalmau, V. (2005). Linear datalog and bounded path duality of relational structures. *Logical Methods in Computer Science*, 1(1).

Immerman, N. (1982). Upper and lower bounds for first order expressibility. *Journal of Computer and System Sciences*, 25(1):76–98.

Immerman, N. (1999). *Descriptive complexity*. Springer-Verlag, New York.

Kolaitis, P. G. and Vardi, M. Y. (1995). On the expressive power of datalog: Tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134.

Libkin, L. (2004). *Elements of finite model theory*. Springer-Verlag, Berlin. Freely available at https://homepages.inf.ed.ac.uk/libkin/fmt/fmt.pdf.

Nesetril, J. and de Mendez, P. O. (2006). Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041.

Robertson, N. and Seymour, P. D. (1986). Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322.