Explorations in Coalgebraic Predicate Logic
With a Focus on Interpolation

MSc Thesis *(Afstudeerscriptie)*

written by

Rover Junior Samwel
(born September 2, 1999 in Amsterdam)

under the supervision of **prof. dr. Yde Venema**, and submitted to the Examinations Board in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
October 31, 2022	dr. Nick Bezhanishvili
	dr. Malvin Gattinger (chair)
	prof. dr. Helle Hvid Hansen
	prof. dr. Yde Venema (supervisor)

Institute for Logic, Language and Computation

Abstract

This thesis explores the area of coalgebraic predicate logic (CPL) in various ways. We describe the morphisms of CPL in great detail, motivating our choices and giving a characterisation in the form of a 'Coalgebraic Diagram'. Furthermore, we establish an Ehrenfeucht Fraïssé (EF) game for CPL, along with an adequacy result. The focus of the thesis is on interpolation results for CPL, with both a semantic and a syntactic approach. Semantically, we generalise the set up of the colimit construction in first-order logic, working towards interpolation via Robinson's consistency. This results in an analysis that explains how colimits can only work in CPL under a tight restriction.

Syntactically, we provide an interpolation result for monotone neighbourhood frames that arise from CPL, using Maehara's method. The proof theoretic portion of the thesis also gives a road map for further interpolation results in CPL. The latter stresses that there is yet a lot to be done in the area of CPL, both in its model theory and its proof theory. The EF-game can for example be further studied, generalised and broken down in variations. And the aforementioned restriction on colimits in CPL as well as the road map for general interpolation results can be further explored.

Keywords— Coalgebraic Predicate Logic, Predicate Liftings, Interpolation, Maehara's method, Ehrenfeucht-Fraïssé games, Monotone Neighbourhood Functor

Contents

1 Acknowledgements

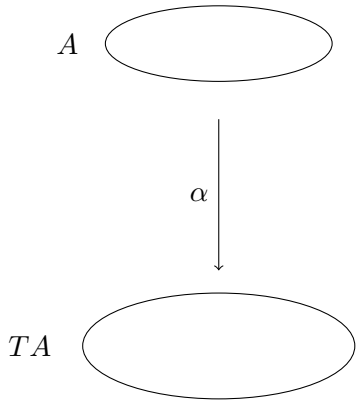
Although this thesis has been a solo-project there are some people I could not have done it without. First of all, I want to thank my supervisor Yde Venema for all his help, patience, insights and motivations. I also want to thank the committee members for their time and effort as well as a defence that felt more like a fun conversation about coalgebraic predicate logic among peers than the strict interrogation that I was anxious for.

I also want to thank Katsuhiko Sano for taking the time to meet with me and talk about CPL. It was very refreshing to talk in-depth about the material with one of the authors of the article that was my main inspiration. Chapter 7 on Maehara's method would not have been there without you.

Much thanks also goes to Evan and Francesco, my MoL study buddies who kept me sharp and motivated. Thank you to my parents and sisters who, despite my warnings that they wouldn't understand anything, all attended my defence. And thank you to all the friends who held the patience to listen to me rambling about CPL and its intricacies, you know who you are. ♡

2 Introduction

Coalgebra is a mathematical framework used to describe state-based systems. Given a set functor T, a T-coalgebra is a set A together with a coalgebra map α to the set TA.



In the case that T is the powerset functor, the coalgebra map α for example assigns each element a set of states in A, in which case we are effectively describing Kripke models. Other examples of coalgebras are automata, computer programs or a vending machine. One might see a coalgebra as the user-perspective of a machine. We view the inside of the machine as a black box and concern us with what happens when we push a button in a certain state of the machine. Given a set functor one can create an unlimited number of different kinds of structures using coalgebras.

Logical approaches of coalgebra have mainly focused on generalising modal logic to coalgebraic modal logic (CML), using relation liftings[1] or *predicate liftings*. Being one of the key players in this thesis, predicate liftings are modal operators in the form of natural transformations; the semantics of these modalities is defined in terms of the natural transformation and the coalgebraic structure of respective models. The interaction between modal logics and coalgebra is for example studied by Leal [2] and Kupke and Pattinson [3].

This thesis concerns itself with the combination of CML and first-order logic (FOL): coalgebraic predicate logic (CPL). The idea to also add first-order quantifiers and predicates to the environment of modal logic - the logic of neighbourhood systems to be precise - originally comes from Chang [4]. However, Litak, Pattinson, Sano and Schröder [5] really established CPL as a discipline, generalising Chang's work to the framework of coalgebras and predicate liftings.

Because it is quite important for the development of the thesis, we will take the time to discuss the syntax, semantics and key results of CPL given by Litak et al. [5], who have set out both the model theory and proof theory of CPL. While doing so, we also take a look at the connection between Chang's logic and its CPL counterpart. Discussing this

[1] Relation liftings will not be discussed in this thesis. For an overview see for example [1].

groundwork will most importantly enable to set up a detailed description of morphisms for CPL, by realising that the semantics is more important than the coalgebraic structure. Moreover, it will give us the chance to generalise some of the ideas from Chang [4] to CPL. In addition, we define a 'Coalgebraic Diagram' that captures all the information of a CPL-model.

The focus of the thesis is on interpolation, which has for CML for example been studied by Seifan, Schröder and Pattinson [6]. We cover two approaches, one model theoretic and one proof theoretic. Our model theoretic approach generalises the colimit construction of FOL, which we will describe in detail to work for specific cases. And our proof theoretic approach uses sequent system of CPL given in [5] by employing Maehara's method. We show interpolation for the logic obtained by the monotone neighbourhood logic and provide a road map for more interpolation results in CPL using the same method.

Another result, somewhat unrelated to other parts of the thesis, is an Ehrenfeucht-Fraïssé (EF) game for CPL. We take inspiration from EF-games for generalised quantifiers and set up a game between CPL-models. The game is proven to be adequate and we stress some of the differences between our game on the one hand and classical EF-games as well as bisimulation games for modal logic on the other hand.

Structure of the Thesis. We start by giving an overview of CPL by going through the main beats of both [5] and [4] in chapter 3. From that basis, we will go through the different degrees of morphisms between models for CPL in chapter 4. We provide the Ehrenfeuch-Fraïssé game for CPL in chapter 5. Chapter 6 then gives our semantic approach towards Craig interpolation, while the 7th and final chapter of the body of the thesis gives the proof theoretic approach for interpolation.

We close of this introduction by stating that we expect the reader to be on the level of a graduate logic student who is familiar with coalgebraic modal logic. For basic definitions of and notational clarifications, one can consult the appendix at the end of the document.

3 Coalgebraic Predicate Logic

In this chapter we establish the basics of CPL, mainly by drawing from the two main inspirations of this thesis: Litak, Pattinson, Sano and Schröder [5] and Chang [4]. We give definitions that form the groundwork for later chapters in this thesis and we cover key results of CPL. The inclusion of Chang's paper gives us a look at the origins of CPL and will allow us to generalise some of Chang's ideas later in the thesis.

As this chapter also goes into the syntax, we now fix an infinite set of individual variables $iVar$ for the rest of the thesis.

3.1 Basics of CPL

Litak et al. [5] give the most extensive account on the model theory and proof theory of CPL so this section mostly follows their article. We most importantly cover the syntax and semantics of CPL. Along the way, we will add some original definitions such as constants for the CPL-syntax. After this, we also provide examples and some key results of [5].

3.1.1 Syntax & Semantics

The syntax of CPL combines those of first-order logic and coalgebraic modal logic. Litak et al. [5] characterise CPL-languages with a pair (Λ, Σ), where Λ is a modal similarity type - a set of modal operators - and Σ is a set of first-order predicates. Both the modalities $\heartsuit \in \Lambda$ and the predicates $P \in \Sigma$ come with finite arities $\mathrm{ar}\heartsuit, \mathrm{ar}P \in \omega$. For reasons that will be made clear in chapter 4 we here add a set of constants Γ, which gives rise to the following definition[2].

Definition 3.1. *Given a CPL-language $(\Lambda, \Sigma, \Gamma)$, t is a term iff $t = x$ for some $x \in iVar$ or $t = c$ for some $c \in \Gamma$.*

We thus move from a pair to a triple $(\Lambda, \Sigma, \Gamma)$ to characterise a CPL-language. Given such a triple, we define the set of CPL$(\Lambda, \Sigma, \Gamma)$-formulas inductively:

$$\varphi ::= s = t \mid P(\vec{t}) \mid \bot \mid \neg\varphi \mid \varphi \to \varphi \mid \forall x.\varphi \mid x\heartsuit\lceil y_1 : \varphi_1\rceil ... \lceil y_n : \varphi_2\rceil \tag{1}$$

where s, t, $\vec{t}\,(= t_1, .., t_k)$ are terms, x, $y_1, .., y_n$ are variables, $P \in \Sigma$ is k-ary, and $\heartsuit \in \Lambda$ is n-ary. We may additionally use the abbreviations $\vee$, $\wedge$ and $\exists$. In the modality clause, y in $\lceil y : \varphi\rceil$ is a comprehension variable so that '$\lceil y : \varphi\rceil$ denotes a subset of the carrier of the model' [5, p. 4]. Just as in FOL, we say that a formula is a sentence if it has no free variables[3] and we call a collection of CPL$(\Lambda, \Sigma, \Gamma)$-sentences a CPL$(\Lambda, \Sigma, \Gamma)$-theory. Furthermore, the following abbreviation will be helpful in chapter 6.

[2] As also noted in [5], one could furthermore add function symbols, which would of course slightly change definition 3.1.

[3] See [5, p. 5] for a detailed account of free variables and substitution in CPL.

Definition 3.2. *We say that* $(\Lambda_0, \Sigma_0, \Gamma_0) \subseteq (\Lambda_1, \Sigma_1, \Gamma_1)$ *when* $\Sigma_0 \subseteq \Sigma_1$, $\Lambda_0 \subseteq \Lambda_1$ *and* $\Gamma_0 \subseteq \Gamma_1$.

As the set-up of the syntax suggests, a CPL-model is in essence a FOL-model with coalgebra structure. Given a set functor T and a CPL-language $(\Lambda, \Sigma, \Gamma)$, consider a T-coalgebra (A, α). We endow the carrier set A with interpretations for the first-order predicates and the constants[4]:

$$I_\Sigma : \Sigma \to \bigcup_{n \in \omega} \mathcal{P}(A^n)$$
$$I_\Gamma : \Gamma \to A$$

respecting the arities of all $P \in \Sigma$ and sending each $c \in \Gamma$ to an element $I_\Gamma(c)$ in A. We usually treat I_Σ and I_Γ as one, writing I to mean either or both. Furthermore, the triple $\mathcal{A} = (A, \alpha, I)$ is called a CPL$(\Lambda, \Sigma, \Gamma)$-model or, if the language is understood, a CPL-model, a coalgebraic model or a T-model. Given an interpretation I we also use $P^{\mathcal{A}}$ and $c^{\mathcal{A}}$ for $I(P)$ and $I(c)$ respectively.

For the coalgebra part, we interpret all $\heartsuit \in \Lambda$ with associated predicate liftings $[|\heartsuit|]$, respecting the arities. So for any CPL$(\Lambda, \Sigma, \Gamma)$-model $\mathcal{A} = (A, \alpha, I)$, given an assignment $v : iVar \to A$, we have

$$\mathcal{A}, v \models x\heartsuit\lceil y_1 : \varphi_1\rceil ... \lceil y_n : \varphi_n\rceil \Leftrightarrow \alpha\big(v(x)\big) \in [|\heartsuit|]_A\big([|\varphi_1(y_1)|]^{y_1}_{\mathcal{A}}, .., [|\varphi_n(y_n)|]^{y_n}_{\mathcal{A}}\big) \tag{2}$$

where

$$[|\varphi(y)|]^y_{\mathcal{A}} = \{a \in A \mid \mathcal{A}, v[a/y] \models \varphi\}. \tag{3}$$

Given a similarity type Λ, we call a set functor together with a predicate lifting $[|\heartsuit|]$ for each modal operator $\heartsuit \in \Lambda$ a Λ-structure[5]. Using Λ-structure makes it easier to talk about instances of CPL without specifying Σ and Γ, which do not depend on the set functor. Furthermore, since the distinction between modal operators and their interpretations is clear, we will from now on use the two terms *modal operator* and *predicate lifting* interchangeably.

For the rest of the syntax: the logical connectives and the first-order quantifiers are interpreted in the usual way. Given a CPL-model $\mathcal{A}$, we denote the theory of all CPL-sentences true in $\mathcal{A}$ as $\mathrm{Th}(\mathcal{A})$.

Because our treatment of variables is exactly the same as in FOL, we can also adopt the following proposition[6].

Proposition 3.3. *Given a* $(\Lambda, \Sigma, \Gamma)$*-theory* T *and a formula* $\varphi(x) \in CPL(\Lambda, \Sigma, \Gamma)$ *in which the constant* c *does not occur, we have*

$$T \models \varphi(c) \Leftrightarrow T \models \forall x \varphi(x). \tag{4}$$

[4]We added I_Γ to accommodate the constants. So our definition of the interpretation slightly differs from the one given in [5].

[5]See also definition 3.1 in [3].

[6]See e.g. [7, p. 14].

This will prove helpful later on. We move on to instances of CPL, giving some examples of Λ-structures.

3.1.2 Examples

The following three examples are also given in [5]. Here we give the semantics of the modal operators and some relevant context. All three will return as running examples.

Relational first-order logic Let $T = \mathcal{P} \times \Delta_{\mathcal{P}\mathrm{At}}$, where At is a set of atomic propositions. Then we get relational FOL[7]. Models of this logic basically are Kripke models in coalgebraic notation. Let $\mathcal{A} = (A, \alpha, I)$ be a $\mathcal{P} \times \Delta_{\mathcal{P}\mathrm{At}}$-model. Each $a \in A$ gets assigned a set $\alpha(a)(1)$ of successors and a set $\alpha(a)(2)$ of atomic propositions that are true at a. The predicate lifting $\Diamond$, given a subset C of A, is defined as follows:

$$[|\Diamond|]_A(C) := \{(X, U) \in \mathcal{P}X \times \mathcal{P}\mathrm{At} \mid C \cap X \neq \emptyset\} \tag{5}$$

Furthermore, the elements of At are also predicate liftings. For any $a \in \mathrm{At}$ we have

$$[|a|]_A(C) := \{(X, U) \in \mathcal{P}X \times \mathcal{P}\mathrm{At} \mid a \in U\}. \tag{6}$$

So we have $\Lambda = \{\Diamond\} \cup At$. As noted above, Σ and Γ can be anything.

To state that a has b as one of its successor, we may write

$$a\Diamond\lceil z : z = b\rceil.$$

Litak et al. [5] give the translation from relational FOL, stating that relational FOL is expressively equivalent to its correspondence language.

Neighbourhood Frames If $T = \mathcal{N} = \mathcal{Q}\mathcal{Q}$, we get neighbourhood frames. Let (A, α) be a $\mathcal{Q}\mathcal{Q}$-coalgebra. Then each $a \in A$ is associated with a set of neighbourhoods, subsets of the carrier set A. The sole predicate lifting is $[|\Box|]$, which is, given some $C \subseteq A$, defined as follows.

$$[|\Box|]_A(C) := \{\alpha \in \mathcal{Q}\mathcal{Q}A \mid C \in \alpha\} \tag{7}$$

We often impose *monotonicity* on the collection of neighbourhoods of an element a:

$$C_0 \in \alpha(a),\ C_0 \subseteq C_1 \Rightarrow C_1 \in \alpha(a)$$

We then have for $C_0 \subseteq C_1$ that $[|\Box|]_A(C_0) \subseteq [|\Box|]_A(C_1)$; the predicate lifting becomes monotone as well. We refer to the monotone neighbourhood functor as $\mathcal{M}$, on which chapter 7 will zoom in. The logic for $\mathcal{N}$ can be axiomatised by[8]:

[7] We here mean the usual correspondence language, a notational variant of FOL [5, p. 7].

[8] This *congruence* rule actually holds for all predicate liftings.

$$\frac{p \leftrightarrow q}{\Box p \to \Box q}$$

Notice that if we work with monotone neighbourhoods, we can drop the right-to-left implication in the premise to get the following rule:

$$\frac{p \to q}{\Box p \to \Box q}$$

We return to the latter rule in chapter 7.

Graded Modal Logics For graded modal logics, we consider the functor $\mathcal{B}$, defined on objects by

$$\mathcal{B}A := \{\mu : A \to \mathbb{N} \cup \{\infty\} \mid \mu \text{ is a map } \}. \tag{8}$$

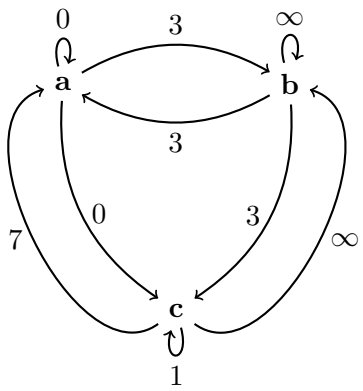
Let (A, α) be a $\mathcal{B}$-coalgebra. A map $\mu : A \to \mathbb{N} \cup \{\infty\}$ may be seen as a measure on A, i.e. giving for each possible 'path' between elements in A a value. For $C \subseteq A$ we write $\mu(C)$ for $\Sigma_{c \in C}\mu(c)$. On morphisms $f : A \to B$, the functor $\mathcal{B}$ takes image measures:

$$\mathcal{B}f(D) := \mu(f^{-1}[D]) \text{ for } D \subseteq B \tag{9}$$

Graded modal logic has infinitely many predicate liftings: $[|\langle k \rangle|]$ for $k \geq 0$, defined by

$$[|\langle k \rangle|]_A(C) := \{\mu \in \mathcal{B}A \mid \mu(C) > k\}, \tag{10}$$

expressing that more than k successors have the property described by C. We may use a picture like below to represent a $\mathcal{B}$-coalgebra intuitively.



To understand the picture, we for example have $\alpha(a)(b) = 3$. Of course, one can change the functor by changing the range of the functions μ, for example to $[0, 1]$ so that we assign probabilities instead.

3.1.3 Key Results

In this section we mention some of the key results found by Litak et al. [5]. We want to give a general overview without worrying too much about technical details.

The most important result is completeness, the proof of which in heavily relies on one-step logic and the notion of boundedness[9].

Definition 3.4. *A modal operator $\heartsuit$ is k-bounded in the i-th argument for $k \in \omega$ and with respect to a Λ-structure T if for every set A and every $C_1, .., C_n \subseteq A$,*

$$[|\heartsuit|]_A(C_1, .., C_n) = \bigcup_{B \subseteq C_i, |B| \leq k} [|\heartsuit|]_A(C_1, .., C_{i-1}, B, C_{i+1}, .., C_n).$$

Note that boundedness in the i-th component implies monotonicity in the i-th component. Boundedness can be captured syntactically with so-called *colourings* to include a bounded-ness axiom in the Hilbert-style calculus $\mathcal{HR}$ developed for CPL[10]. Using a Henkin-style construction with witnesses, [5] provides a completeness result for CPL that depends on the one-step rules necessary for the system $\mathcal{HR}$.

Now is a good time to mention that if a Λ-structure makes a modal operator $\heartsuit \in \Lambda$ ω-bounded (by finite sets) but not k-bounded for finite k, strong completeness fails [5, p. 26]. Furthermore,'the bounded structures are the only ones that satisfy compactness'[5, p. 25]. This is compactness (in the 'normal' sense) of CPL-formulas (although the paper does not explicitly state this).

Other than completeness, the model theory of [5] also a CPL-version of the Omitting Types Theorem, which is proved in a similar manner as completeness. Furthermore, [5] explores CPL-ultraproducts (with a version of Lös's theorem) and achieves a Downward Löwenheim-Skolem theorem. The paper extends its reach by also covering the correspondence between CPL and CML, giving translations as well as preservation theorems. They also dive into hybrid languages and other correspondences which are all very interesting but not relevant for the current thesis.

In the proof-theory part of [5] a cut-free complete sequent calculus for CPL is given, which we will study a bit more in chapter 7. For now, it suffices to know that a sound and (under restrictions) complete sequent system for CPL is developed.

As a last - but certainly not least - key result we want to mention the companion paper [8] to [5], in which a version of the Van Benthem/Rosen theorem is proven for CPL. The proof and the machinery used to get there is quite interesting and worth to take a look at.

[9] Definition 3.4 is definition 3.12 in [5, p. 15].

[10] See [5, p. 17] for the details.

3.2 Chang: Modal Model Theory

The ideas put forth in [5] are largely based on earlier[11] work by Chang [4]; the logic described there can be seen as an instance of CPL. As we will use some ideas from Chang's paper later on, especially in chapter 6, it is worthwhile to cover what he established. It is notable that Chang is also concerned with the philosophical and social analysis behind the motivation of the semantics of his logic. In this section, we first give the syntax and semantics and then the motivation behind the semantics.

3.2.1 Syntax and Semantics

Let us first have a look at Chang's syntax and semantics. The logic that Chang defines is a notational variant of Neighbourhood CPL (as described above).

Syntax We have a language L consisting of predicates, functions, individual constants, the identity symbol, individual variables and *modal operators* $N, M, ..$ with arities $ar(N), ar(M), .. \in \omega$. Chang's treatment of terms, first-order predicates and logical connectives and quantifiers is as usual; the most important construction clause is:

(iv) if N is an n-place modal operator, x is a variable, and $\varphi_1, .., \varphi_n$ are formulas then $Nx(\varphi_1 \dots \varphi_n)$ is a formula [4, p. 2]

This clause 'replaces' the last clause in (1). Notice that there is no binding variable in the modal formulas. However, φ_i, for $1 \leq i \leq n$, is meant to denote a subset of the domain.

Semantics Chang uses Gothic letters (e.g. $\mathfrak{A}, \mathfrak{B}$) for his models. We will refer to Chang's models as *neighbourhood* models (not coalgebraic models). A neighbourhood model $\mathfrak{A} = (A, ...)$[12] for a language L consists of a domain A together with interpretations (...) for all symbols in the language L [4, p. 7]. Let N be an n-ary modal operator. The interpretation of N in a model $\mathfrak{A}$, denoted as $N^{\mathfrak{A}}$, is a function that maps every $a \in A$ to a subset of $\mathcal{P}(A) \times \dots \times \mathcal{P}(A)$ (n times). Now let $a_1, .., a_n \in A$ and, for $1 \leq i \leq n$, let $\varphi_i(x_1, .., x_n)$ be a formula. The semantics of N is defined by

$$\mathfrak{A} \models Nx_1(\varphi_1, .., \varphi_m)[a_1, a_2, .., a_n] \Leftrightarrow \\ (\{c_1 \in A \mid \mathfrak{A} \models \varphi_1[c_1, a_2, .., a_n]\}, .., \{c_m \in A \mid \mathfrak{A} \models \varphi_m[c_m, a_2, .., a_n]\}) \in N(a_1). \quad (11)$$

Notice how Chang's treatment of variables slightly differs from the notation used in the rest of this thesis; e.g. a_1 is the assignment of x_1 and the first free variable in φ_1 is equivalent to the binding variable in the modality clause for CPL.

[11] The paper was published in the 70s.

[12] The dots are Chang's own notation.

Let us now see how we can via *transformations* go back and forth between coalgebraic models (for $T = \mathcal{QQ}$) and neighbourhood models. While doing this, we will drop the interpretation of the 'ordinary' predicates Σ and constants Γ; the transformation is really between Chang's modal operators and CPL's predicate liftings. Observe that CPL is quite efficient in the sense that we can define as many predicate liftings as we would like given a set-functor. However, with Chang's neighbourhood models *each* modal operator has its own successor structure.

Chang to CPL We only treat the case that L has finitely many modal operators, all unary. Let $\mathfrak{A} = (A, N_1^{\mathfrak{A}}, .., N_n^{\mathfrak{A}})$ be a neighbourhood model. Each $N_i^{\mathfrak{A}}$ is a function $A \to \mathcal{PP}A$. So we take the functor $T = \mathcal{QQ} \times .. \times \mathcal{QQ}$ (n times) so that a T-coalgebra has a map that for each $a \in A$ gives an n-tuple of neighbourhood-systems. We define $\mathcal{A} = (A, \alpha)$ as the *coalgebraic version* of $\mathfrak{A}$, where α combines all information of $N_i^{\mathfrak{A}}$, for $1 \leq i \leq n$: i.e.

$$\alpha(a)(i) = N_i^{\mathfrak{A}}(a) \text{ for all } a \in A$$

Then we have predicate liftings $[|\Box_i|]$, for $1 \leq i \leq n$, defined in the usual way, just amended to having multiple sets of neighbourhoods:

$$[|\Box_i|]_A(C) = \{\alpha \in \mathcal{QQ}A \times \cdots \times \mathcal{QQ}A \mid C \in \alpha(i)\}$$

for all A, $C \subseteq A$. Slightly abusing notation, we thus have, again for $1 \leq i \leq n$:

$$\begin{aligned} \mathfrak{A} \models aN_i(C) &\Leftrightarrow C \in N^{\mathfrak{A}}(a) \\ &\Leftrightarrow C \in \alpha(a)(i) \\ &\Leftrightarrow \alpha(a) \in [|\Box_i|]_A(C) \\ &\Leftrightarrow \mathcal{A} \models a\Box_i(C) \end{aligned}$$

A syntactic translation from Chang to CPL would thus translate $aN_i(\varphi)$ to $a\Box_i\lceil y : \varphi\rceil$.

CPL to Chang Let $\mathcal{A} = (A, \alpha)$ be a $\mathcal{QQ}$-model. Given the predicate lifting $[|\Box|]$, define

$$N^{\mathfrak{A}} : \ a \mapsto \alpha(a) \in \mathcal{QQ}A \text{ for all } a \in A.$$

Let us call $\mathfrak{A} = (A, N^{\mathfrak{A}})$ the *Chang-version* of $\mathcal{A}$. We have

$$\begin{aligned} \mathcal{A} \models a\Box(C) &\Leftrightarrow \alpha(a) \in [|\Box_i|]_A(C) \\ &\Leftrightarrow C \in \alpha(a)(i) \\ &\Leftrightarrow C \in N_i^{\mathfrak{A}}(a) \\ &\Leftrightarrow aN(C). \end{aligned}$$

It is quite straightforward to see that in the case that we only have one modal operator (which is always for CPL neighbourhood frames) these two *transformations* are each other's inverses.

3.2.2 Motivation

As we may conclude from the above, Chang's semantics is more or less the same as the semantics for the neighbourhood instance of CPL. However, Chang has an inner motivation for this semantics and as we will see, it is a simplification of multiple factors. Here follows an overview.

Given an n-place modal operator N and formulas $\varphi_1, .., \varphi_n$, Chang suggests to read $xN\varphi_1 \dots \varphi_n$ as 'x finds it N that $\varphi_1, .., \varphi_n$' [4, p. 3], where N might be replaced by words such as *possible*, *likely* or *strange*. Chang leaves open what $\varphi_1, .., \varphi_n$ should represent. He suggests a conjunction but does make the distinction between something of the form $xN\varphi_1..\varphi_n$ and something of the form $xM\varphi_1 \wedge .. \wedge \varphi_n$, where M is unary. To be able to analyse the meaning of $xN\varphi_1 \dots \varphi_n$ more efficiently, Chang gives an example: the binary predicate P, which means 'talking to'. The meaning of $aNP(ab)$ should, following [4], depend on at least four factors:

$$a,\ b,\ P(xy) \text{ and } C := \{c \in A \mid Pcb\}.$$

However, this is for simplicity's sake narrowed down to only a and the set C, stating that

> .. individuals are making some sort of judgement in their world, namely $\mathfrak{A}$, based upon information supplied by the interpretation of N in $\mathfrak{A}$. [4, p. 4]

From here comes the definition of N as a function and the semantics of $N^{\mathfrak{A}}$ as described above. Chang admits the simplification and

> .. [leaves] open and untouched the possibility of more complicated interpretations of N. [4, p. 5]

This possibility is not one that the current thesis sets out to dive into but it is at least interesting that coalgebraic models also support his motivation.

The narrowing down of the factors to just a and C leads to the following validity, capturing the congruence of the modal operators.

$$\forall x(\varphi(x) \leftrightarrow \psi(x)) \rightarrow \forall x(Nx\varphi(x) \leftrightarrow Nx\psi(x))$$

In CPL we have the same:

$$\forall y(\varphi(y) \leftrightarrow \psi(y)) \rightarrow \forall x(x\heartsuit\lceil y : \varphi(y)\rceil \leftrightarrow x\heartsuit\lceil y : \psi(y)\rceil)$$

Notice the different variables due to the binding variable in modalities in CPL.

It is worth mentioning that Chang's paper covers some important model theoretic results and constructions for neighbourhood models. Results include the Tarski-Vaught test, compactness and the Löwenheim-Skolem theorem; constructions include co-chains, which we will reference in Chapter 6. One of the questions that the paper ends with will be (partially) answered in chapter 7, where we prove interpolation for *monotone* neighbourhood CPL. A bit more on this paper will follow in the next chapter where we discus the morphisms between coalgebraic models and start adding to and building on the work discussed in this chapter.

4 Morphisms of CPL

This chapter introduces the basic notions of morphisms between CPL-models while also discussing why certain conditions are left out. Section 4.1 brings us all the way to elementary CPL-embeddings, section 4.2 discusses a *coalgebraic condition* introduced by Chang that will give some insight into the nature of coalgebraic models and section 4.3 generalises the Diagram-method of model theory for first-order logic to CPL. Most of this chapter is a generalisation of first-order logic but it is good to record these definitions and results as we will use them later on.

4.1 CPL-Morphisms

For the rest of the chapter, fix a set-functor T and a CPL-language $(\Lambda, \Sigma, \Gamma)$.

In [5] the only remark about morphisms between coalgebraic models comes after the statement of the Löwenheim-Skolem theorem:

> .. we use the term *elementary substructure* in the usual way to designate first-order substructures whose elements satisfy the same formulas as they do in the original model: we explicitly do *not* require that the coalgebra structure on the substructure forms a subcoalgebra. [5, p. 33]

From this, we take the cue that all kinds of CPL-morphisms should be defined in a way that is similar to the corresponding definitions in model theory for first-order logic. That is, focusing on satisfaction and interpretation rather than coalgebraic structure.

Definition 4.1. *Given two CPL$(\Lambda, \Sigma, \Gamma)$-models $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$, a map $f : A \to B$ is a strong CPL$(\Lambda, \Sigma, \Gamma)$-morphism when the following hold:*

1. *For all constants $c \in \Gamma$; $f(c^{\mathcal{A}}) = c^{\mathcal{B}}$.*
2. *For all $a_1, .., a_n \in A$ and all n-ary $P \in \Sigma$;*

$$(a_1, .., a_n) \in I(P) \Leftrightarrow (f(a_1), .., f(a_n)) \in J(P)$$

If f is also injective then f is a CPL$(\Lambda, \Sigma, \Gamma)$-embedding.

It is not hard to show that, given coalgebraic models $\mathcal{A}$ and $\mathcal{B}$, having a CPL$(\Lambda, \Sigma, \Gamma)$-embedding from $\mathcal{A}$ to $\mathcal{B}$ is equivalent to $\mathcal{A}$ and $\mathcal{B}$ making true the same quantifier- *and* modality-free CPL$(\Lambda, \Sigma, \Gamma)$-formulas.

Definition 4.2. *A CPL$(\Lambda, \Sigma, \Gamma)$-embedding $f : A \to B$ between T-models $\mathcal{A}$ and $\mathcal{B}$ is elementary if we have*

$$\mathcal{A} \models \varphi(a_1, .., a_n) \Leftrightarrow \mathcal{B} \models \varphi(f(a_1), .., f(a_n))$$

for all $a_1, .., a_n \in A$ and all CPL$(\Lambda, \Sigma, \Gamma)$-formulas $\varphi(x_1, .., x_n)$.

If in a certain context a CPL-language $(\Lambda, \Sigma, \Gamma)$ is understood, we may state e.g. CPL-embedding instead of CPL$(\Lambda, \Sigma, \Gamma)$-embedding. Note that nothing in definitions 4.1 or 4.2 says anything about the coalgebra structure of CPL-models. However, the next proposition captures the strength of a coalgebra morphism in terms of preservation of truth.

Proposition 4.3. *Let $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$ be two CPL$(\Sigma, \Lambda, \Gamma)$-models. If a CPL$(\Sigma, \Lambda, \Gamma)$-embedding $f : A \to B$ is a coalgebra morphism then*

$$\mathcal{A} \models \varphi(a_1, .., a_n) \Leftrightarrow \mathcal{B} \models \varphi(f(a_1), .., f(a_n))$$

for all $a_1, .., a_n \in A$ and all quantifier-free CPL$(\Sigma, \Lambda, \Gamma)$-formulas $\varphi(x_1, .., x_n)$.

Proof. Given the triviality of the atomic and Boolean cases, we only cover the modality case. Recall that the assumption that f is a coalgebra morphism means that

$$\beta \circ f = Tf \circ \alpha. \tag{12}$$

Now let

$$\varphi(x_1, .., x_n) = x_1 \heartsuit \ldots \lceil y_i : \psi_i(x_2, .., x_n, y_i) \rceil \ldots \tag{13}$$

for some k-ary $\heartsuit \in \Lambda$ and arbitrary CPL$(\Lambda, \Sigma, \Gamma)$-formulas $\psi_i(x_2, .., x_n, y_i)$ for $1 \leq i \leq k$, for which the induction hypothesis yields

$$a \in [|\psi_i(a_2, .., a_n, y_i)|]^{y_i}_{\mathcal{A}} \Leftrightarrow f(a) \in [|\psi_i(f(a_2), .., f(a_n), y_i)|]^{y_i}_{\mathcal{B}} \tag{14}$$

for all $a, a_2, .., a_n \in A$. If we for a moment fix arbitrary $a_2, .., a_n \in A$, the naturality condition for $[|\heartsuit|]$ with respect to f, starting with $[|\psi(f(a_2), .., f(a_n), y)|]^y_{\mathcal{B}} \in \mathcal{Q}B$ gives

$$\begin{aligned}[|\heartsuit|]_A(.., [|\psi_i(a_2, .., a_n, y_n)|]^{y_i}_{\mathcal{A}}, ..) = \\ \{\alpha \in TA \mid Tf(\alpha) \in [|\heartsuit|]_B(.., [|\psi_i(f(a_2), .., f(a_n), y_i)|]^{y_i}_{\mathcal{B}}, ..).\end{aligned} \tag{15}$$

We now show that $\mathcal{A} \models \varphi(a_1, .., a_n) \Leftrightarrow \mathcal{B} \models \varphi(f(a_1), .., f(a_n))$ for all $a_1, .., a_n \in A$. Take arbitary $a_1, .., a_n \in A$. We have

$$\begin{aligned}
&\mathcal{A} \models a_1 \heartsuit .. \lceil y_i : \psi(a_2, .., a_n, y_i) \rceil .. \\
&\quad \Rightarrow \alpha(a_1) \in [|\heartsuit|]_A(.., [|\psi_i(a_2, .., a_n, y_i)|]^{y_i}_{\mathcal{A}}, ..) && \text{(definition)} \\
&\quad \Rightarrow Tf\big(\alpha(a_1)\big) \in [|\heartsuit|]_B(.., [|\psi_i(f(a_2), .., f(a_n), y_i)|]^{y_i}_{\mathcal{B}}, ..) && \text{(by (15))} \\
&\quad \Rightarrow \beta(f(a_1)) \in [|\heartsuit|]_B(.., [|\psi_i(f(a_2), .., f(a_n), y_i)|]^{y_i}_{\mathcal{B}}, ..) && \text{(by (12))} \\
&\quad \Rightarrow \mathcal{B} \models f(a_1) \heartsuit .. \lceil y_i : \psi(f(a_2), .., f(a_n), y_i) \rceil ... && \text{(definition).}
\end{aligned}$$

We thus conclude that $\mathcal{B} \models f(a_1) \heartsuit \ldots \lceil y_i : \psi(f(a_2), .., f(a_n), y_i) \rceil \ldots$. The implications actually go both ways so the proof is done. □

Definition 4.4. *Two CPL$(\Lambda, \Sigma, \Gamma)$-models $\mathcal{A}$ and $\mathcal{B}$, are $(\Lambda, \Sigma, \Gamma)$-equivalent, write $\mathcal{A} \equiv_{(\Lambda,\Sigma,\Gamma)} \mathcal{B}$, when both models make true exactly the same CPL$(\Lambda, \Sigma, \Gamma)$-sentences.*

It is not hard to show that $\equiv_{(\Lambda,\Sigma,\Gamma)}$ is an equivalence relation.

Proposition 4.5. *Let $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$ be two CPL$(\Lambda, \Sigma, \Gamma)$-models. If there is an elementary CPL$(\Lambda, \Sigma, \Gamma)$-embedding $f : A \to B$ then $\mathcal{A} \equiv_{(\Lambda,\Sigma,\Gamma)} \mathcal{B}$.*

Proof. Trivial by the definition (4.2) of an elementary CPL-embedding. □

4.2 A Coalgebraic Condition

Although we will in this thesis stick to the definition of a CPL-morphism given in the previous section, *without* any coalgebraic conditions or assumptions, it is worthwhile to look at the possibilities in that area. In fact, Chang [4] adds a condition to the successor-structure of his neighbourhood models. This section examines this condition and formulates a general version of it, in the coalgebraic setting.

Chang [4] introduces the notion of a *submodel*: A neighbourhood-model $\mathfrak{A} = (A, ...)$[13] is a submodel of $\mathfrak{B} = (B, ...)$ if $\mathfrak{A}$ is a submodel of $\mathfrak{B}$ in the first-order sense[14] *and* for each n-place operator N, each $a \in A$ we have that if $\langle C_1, \ldots, C_n \rangle \in N^{\mathfrak{A}}(a)$, where $C_i \subseteq A$ for $1 \leq i \leq n$, then there are $D_i \subseteq B$, for $1 \leq i \leq n$, such that the following are true:

(i) $C_i = A \cap D_i$ for $1 \leq i \leq n$.

(ii) $\langle D_1, \ldots, D_n \rangle \in N^{\mathfrak{B}}(a)$

For (i), we say that 'D_i extends C_i into B'. Let us now *translate* this condition to the coalgebraic setting, as described in Chapter 3. Let $\mathcal{A} = (A, \sigma^{\mathcal{A}}, I)$ and $\mathcal{B} = (B, \sigma^{\mathcal{B}}, J)$ be the coalgebraic versions of $\mathfrak{A}$ and $\mathfrak{B}$ respectively and let $\Box$ be the coalgebraic version of N. In the case that N is unary,

$$C \in N^{\mathfrak{A}}(a) \text{ translates to } \sigma^{\mathcal{A}}(a) \in [|\Box|]_A(C).$$

Furthermore, if there is a $D \subseteq B$ that extends C into B while also satisfying (ii), then we have $\sigma^{\mathcal{B}}(a) \in [|\Box|]_B(D)$. Note that we can generalise the condition to the case that we do not have $A \subseteq B$. We then want, given a map $f : A \to B$, that $\mathfrak{A}' = (f[A], ...)$ is a submodel of $\mathfrak{B}$. Keeping this in mind, we get the following *coalgebraic condition* for a general Λ-structure.

Definition 4.6. *Given a Λ-structure (T, Λ) and two T-coalgebras (A, α) and (B, β), we say that $f : A \to B$ is a Chang-Λ-morphism if for all n-ary $\heartsuit \in \Lambda$, for all $a \in A$ and for all $C_i \subseteq A$, for $i \leq n$, such that $\alpha(a) \in \heartsuit_A(C_1, .., C_n)$, there are $D_i \subseteq B$ such that:*

[13] Recall that the dots are Chang's notation, as introduced in chapter 3.

[14] Conform the two conditions given in definition 4.1 but Chang [4] does not specify.

(I) $f[C_i] = f[A] \cap D_i$ *for* $i \leq n$

(II) $\beta\big(f(a)\big) \in [|\heartsuit|]_B(D_1, .., D_n)$

If the Λ-structure is understood we just write Chang-morphism. And if f is a Chang-morphism, we alternatively say that f satisfies the Chang-condition.

Chang [4] adds the Chang-condition to his definition of an elementary substructure and also describes the notion of a *conservative* submodel, where the implication goes both ways. Let us now check some basic properties of Chang-morphism.

Proposition 4.7. *Let* (T, Λ) *be a* Λ*-structure and assume that* Λ *consists of monotone predicate liftings only. Then any* T*-coalgebra morphism is a Chang-*Λ*-morphism.*

Proof. Let (A, α) and (B, β) be two T-coalgebras and assume that $f : A \to B$ is a coalgebra morphism. For simplicity we look at unary predicate liftings but this can easily be generalised. Take arbitrary $a \in A$, some $\heartsuit \in \Lambda$ and any $C \subseteq A$.

Now assume that $(\star)$ $\alpha(a) \in [|\heartsuit|]_A(C)$. Consider $f[C] \subseteq B$. Notice that $C \subseteq (f^{-1} \circ f)[C]$ so by monotonicity of $\heartsuit$ and $(\star)$ we have $(*)$ $\alpha(a) \in \heartsuit_A((f^{-1} \circ f)[C])$. Now by naturality of $\heartsuit$ with respect to f we have that

$$[|\heartsuit|]_A\big((f^{-1} \circ f)[C]\big) = \{\alpha \in TA \mid Tf\alpha \in \heartsuit_B(f[C])\}. \tag{16}$$

so by $(*)$ and (16) we have $Tf\big(\alpha(a)\big) \in [|\heartsuit|]_B(f[C])$. Furthermore, f is a coalgebra morphism so $\beta\big(f(a)\big) \in \heartsuit_B(f[C])$. Obviously, $f[C] = f[A] \cap f[C]$ so the condition is satisfied. □

Proposition 4.8. *Let* $\mathcal{A} = (A, \alpha)$, $\mathcal{B} = (B, \beta)$ *and* $\mathcal{E} = (E, \epsilon)$ *be* T*-coalgebras for some set-functor* T *and assume* $f : A \to B$ *and* $g : B \to E$ *are Chang-morphisms. Then* $g \circ f : A \to E$ *is a Chang-morphism as well.*

Proof. Take some $a \in A$, some modal operator $\heartsuit$ (interpreted by the predicate lifting $[|\heartsuit|]$) and some $C \subseteq A$ and assume $\alpha(a) \in [|\heartsuit|]_A(C)$. Then there is some $D \subseteq B$ such that

$$f[C] = f[A] \cap D \text{ and } \beta\big(f(a)\big) \in [|\heartsuit|]_B(D). \tag{17}$$

By the fact that $\beta\big(f(a)\big) \in [|\heartsuit|]_B(D)$, there is some $F \subseteq E$ such that

$$g[D] = g[B] \cap F \text{ and } \epsilon\big(g(f(a))\big) \in [|\heartsuit|]_E(F). \tag{18}$$

Then we have

$$(g \circ f)[C] = (g \circ f)[A] \cap g[D] = (g \circ f)[A] \cap g[B] \cap F \tag{19}$$

and since $(g \circ f)[A] \subseteq g[B]$ we have

$$(g \circ f)[C] = (g \circ f)[A] \cap F. \tag{20}$$

So $g \circ f$ satisfies the Chang-condition. □

As natural as the condition for being a Chang-morphism might seem, it does not guarantee preservation of the truth of quantifier-free $\mathrm{CPL}(\Lambda, \Sigma, \Gamma)$-formulas. Chang [4] also does not claim as much but also does not say the opposite. Let us here explain why this preservation does not work.

Let $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$ be two $(\Lambda, \Sigma, \Gamma)$-models. Furthermore, let the CPL-embedding $f : A \to B$ be a Chang-morphism. Now assume $\mathcal{A} \models a \heartsuit \lceil y : \psi(y) \rceil$ for $a \in A$, $\heartsuit \in \Lambda$ and quantifier-free $\psi(y) \in \mathrm{CPL}(\Lambda, \Sigma, \Gamma)$. By definition $\alpha(a) \in [|\heartsuit|]_A([|\psi(y)|]^y_{\mathcal{A}})$. Since f is a Chang-morphism, there is some $D \subseteq B$ such that

$$\begin{gathered} f\big[[|\psi(y)|]^y_{\mathcal{A}}\big] = f[A] \cap D \\ \text{and } \beta\big(f(a)\big) \in [|\heartsuit|]_B(D). \end{gathered}$$

The 'problem' is that it is not guaranteed that D is the same as $[|\psi(y)|]^y_{\mathcal{B}}$ (or in the case that $\heartsuit$ is monotone, a subset of $[|\psi(y)|]^y_{\mathcal{B}}$). So it is not necessarily the case that $\mathcal{B} \models f(a) \heartsuit \lceil y : \psi(y) \rceil$. Actually, here is an example with $T = \mathcal{QQ}$ to showcase this.

Example 4.9. *Let $T = \mathcal{N}$ with $\Lambda = \{\Box\}$ and take $\Sigma = \{P\}$ with P unary (and $\Gamma = \emptyset$). Define $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$ as follows:*

$$\begin{aligned} A &= \{a_1, a_2\} \\ \alpha(a_1) &= \big\{\{a_2\}\big\}, \alpha(a_2) = \emptyset \\ I(P) &= \{a_2\} \\ B &= \{b_1, b_2, b_3\} \\ \beta(b_1) &= \big\{\{b_2, b_3\}\big\}, \beta(b_2) = \beta(b_3) = \emptyset \\ J(P) &= \{b_2\} \end{aligned}$$

Furthermore, define a map $f : A \to B$:

$$f : a_1 \mapsto b_1, a_2 \mapsto b_2$$

The Chang-condition is satisfied by f; only $\alpha(a_1)$ is of importance and we have

$$f[\{a_2\}] = f[A] \cap \{b_2, b_3\}.$$

However, we have $\mathcal{A} \models a_1 \heartsuit \lceil y : P(y) \rceil$ and $\mathcal{B} \not\models b_1 \heartsuit \lceil y : P(y) \rceil$.

This failed capture of truth-preservation should not be a concern; the most important morphisms are elementary CPL-embeddings and those preserve truth by definition.

To close this section on the Chang-condition, we offer a slightly different version of it. The syntax of CPL is not powerful enough to talk about any subset of a a certain domain so it makes sense to introduce a weaker version of the Chang-morphism as follows.

Definition 4.10. *A map $f : A \to B$ is a weak Chang-morphism if for all predicate liftings $\heartsuit$, all $a_1, .., a_n$ and all $\varphi(x_2, .., x_n, y) \in CPL(\Lambda, \Sigma, \Gamma)$ such that $\alpha(a_1) \in \heartsuit([\varphi(a_2, .., a_n, y)]^y_{\mathcal{A}})$, there is a $D \subseteq B$ such that:*

- $f\big[[\varphi(a_2, .., a_n, y)]^y_{\mathcal{A}}\big] = f[A] \cap D$
- $\beta(f(a)) \in \heartsuit_B(D)$

It should be clear when a map is a conservative weak Chang-morphism.

Proposition 4.11. *Let $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$ be two CPL$(\Lambda, \Sigma, \Gamma)$-models. If $f : A \to B$ is an elementary CPL-embedding, then it satisfies the weak conservative Chang-condition.*

Proof. Take arbitrary $a_1, .., a_n \in A$, $\heartsuit \in \Lambda$ and $\varphi(x_2, .., x_n, y) \in \mathrm{CPL}(\Lambda, \Sigma, \Gamma)$ such that

$$\alpha(a_1) \in [|\heartsuit|]([|\varphi(a_2, .., a_n, y)|]^y_{\mathcal{A}}).$$

Then $\mathcal{A} \models a_1 \heartsuit \lceil y : \varphi(a_2, .., a_n, y) \rceil$, so $\mathcal{B} \models f(a_1) \heartsuit \lceil y : \varphi(f(a_2), .., f(a_n), y) \rceil$ so

$$\beta(f(a_1)) \in [|\heartsuit|]([|\varphi(f(a_2), .., f(a_n), y)|]^y_{\mathcal{B}})$$

and furthermore

$$f\big[[|\varphi(a_2, .., a_n, y)|]^y_{\mathcal{A}}\big] = f[A] \cap [|\varphi(f(a_2), .., f(a_n), y)|]^y_{\mathcal{B}}.$$

So we know that f satisfies the conservative weak Chang-condition. □

In essence, weak Chang-morphisms only look at *CPL-definable* subsets of a domain.

4.3 The Coalgebraic Diagram

The (elementary) Coalgebraic Diagram (ElCoDiag), similar to the first-order (elementary) Diagram[15], is a set of sentences designed to capture all the information in a coalgebraic model, relative to a language. We will show that if some coalgebraic model $\mathcal{B}$ makes true the ElCoDiag of another coalgebraic model $\mathcal{A}$, then there is an elementary CPL-embedding from $\mathcal{A}$ to $\mathcal{B}$. Since we will add names for all elements in the carrier-set of a coalgebraic model when defining its ElCoDiag, this is the chapter where it pays off to allow constants in our language.

Definition 4.12. *Given a CPL-model $\mathcal{A} = (A, \alpha, I)$, we denote the set of individual constants naming all elements in A, the naming constants of $\mathcal{A}$, as*

$$\Gamma(A) := \{c_a \mid a \in A\}.$$

[15] See for example [9, p. 45].

Note that the naming constants of a model do not depend on the CPL-language. Given $\Gamma(A)$ for a model $\mathcal{A}$, we add the natural interpretation $c_a^{\mathcal{A}} = a$. We may denote the expansion of $\mathcal{A}$ that interprets all naming constants in the natural way as $\mathcal{A}^+$.

Definition 4.13. *Let $\mathcal{A} = (A, \alpha, I)$ be a CPL$(\Lambda, \Sigma, \Gamma)$-model. The CoDiag of $\mathcal{A}$ relative to $(\Lambda, \Sigma, \Gamma)$ is the set of all quantifier-free CPL$(\Lambda, \Sigma, \Gamma \cup \Gamma(A))$-sentences that are true in $\mathcal{A}$:*

$$\mathit{CoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A}) := \{\textit{quantifier-free sentences } \varphi \in \mathit{CPL}(\Lambda, \Sigma, \Gamma \cup \Gamma(A)) \mid \mathcal{A} \models \varphi\}.$$

Furthermore:

$$\mathit{ElCoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A}) := \{\textit{sentences } \varphi \in \mathit{CPL}(\Lambda, \Sigma, \Gamma \cup \Gamma(A)) \mid \mathcal{A} \models \varphi\}.$$

Notice that for any CPL$(\Lambda, \Sigma, \Gamma)$-model $\mathcal{A}$ we always have $\mathrm{CoDiag}_{(\Lambda,\Sigma,\Gamma)} \subset \mathrm{ElCoDiag}_{(\Lambda,\Sigma,\Gamma)}$. In the rest of the chapter we will for notational clarity assume that all $\heartsuit \in \Lambda$ are unary. Before we go on to the main theorem, we prove a helpful lemma about the (non-elementary) CoDiag.

Lemma 4.14. *Let $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$ be CPL$(\Lambda, \Sigma, \Gamma)$-models. If $\mathcal{B}$ has a CPL$(\Lambda, \Sigma, \Gamma \cup \Gamma(A))$-expansion $\mathcal{B}^+ = (B, \beta, J^+)$ such that*

$$\mathcal{B}^+ \models \mathit{CoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A})$$

then there is a CPL$(\Lambda, \Sigma, \Gamma)$-embedding $h : A \to B$.

Proof. Assume that $\mathcal{B}$ has a CPL$(\Lambda, \Sigma, \Gamma \cup \Gamma(A))$-expansion $\mathcal{B}^+ = (B, \beta, J^+)$ such that $\mathcal{B}^+ \models \mathrm{CoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A})$. It is to show that there is a CPL$(\Lambda, \Sigma, \Gamma)$-embedding $h : A \to B$. Noticing that $\mathcal{B}^+$ has interpretations of all constants $c_a \in \Gamma(A)$, we define h by

$$h : a \mapsto J^+(c_a) \text{ for all } a \in A. \tag{21}$$

First, we check that h is a strong CPL$(\Lambda, \Sigma, \Gamma)$-morphism by checking the two clauses in definition 4.1.

1. The constants c_a for some $a \in A$ are added to the language for the CoDiag; they are not in the language that h will be shown to be an embedding for.

   For $c \in \Gamma$ with $c^{\mathcal{A}} = a$ for some $a \in A$, we have $c = c_a \in \mathrm{CoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A})$. So $\mathcal{B}^+ \models c = c_a$. Then $c^{\mathcal{B}} = c_a^{\mathcal{B}^+} = J^+(c_a) = h(a)$. So then $h(c^{\mathcal{A}}) = h(a) = c^{\mathcal{B}}$, as desired.

2. For any $a_1, .., a_n$ and any n-ary $P \in \Sigma$ we have

$$\begin{aligned}
(a_1, .., a_n) \in I(P) &\Leftrightarrow \mathcal{A} \models P(c_{a_1}, .., c_{a_n}) && \text{(definition)} \\
&\Leftrightarrow P(c_{a_1}, .., c_{a_n}) \in \mathrm{CoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A}) && \text{(definition 4.13)} \\
&\Leftrightarrow \mathcal{B}^+ \models P(c_{a_1}, .., c_{a_n}) && \text{(assumption)} \\
&\Leftrightarrow (J^+(c_{a_1}), .., J^+(c_{a_n})) \in J^+(P) && \text{(definition)} \\
&\Leftrightarrow (h(a_1), .., h(a_n)) \in J(P) && \text{(by (21))}
\end{aligned}$$

The step from J^+ to J in the last equivalence is justified by the fact that for $P \in \Sigma$ we have $J^+(P) = J(P)$, since $\mathcal{B}^+$ is an expansion. So $\mathcal{B}^+$ interprets already interpreted predicates in the same way as $\mathcal{B}$.

Second, we show injectivity. Take $a_1, a_2 \in A$ and assume $a_1 \neq a_2$. Then $\neg(c_{a_1} = c_{a_2}) \in \mathrm{CoDiag}_{(\Sigma,\Lambda)}(\mathcal{A})$ so $\mathcal{B}^+ \models \neg(c_{a_1} = c_{a_2})$. So $J^+(c_{a_1}) \neq J^+(c_{a_2})$. By Definition 22 we have $h(a_1) \neq h(a_2)$.

We conclude that h is a CPL-embedding. □

Notice that the main theorem below, as opposed to the lemma above, *does* state an equivalence, instead of just one implication.

Theorem 4.15. *Let Λ be a set of monotone predicate liftings. The following are equivalent for two CPL$(\Lambda, \Sigma, \Gamma)$-models $\mathcal{A} = (A, \alpha, I)$ and $\mathcal{B} = (B, \beta, J)$.*

1. *$\mathcal{B}$ has a CPL$(\Lambda, \Sigma, \Gamma \cup \Gamma(A))$-expansion $\mathcal{B}^+ = (B, \beta, J^+)$ such that*
$$\mathcal{B}^+ \models \mathit{ElCoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A}).$$
2. *There is an elementary CPL$(\Lambda, \Sigma, \Gamma)$-embedding $h : A \to B$.*

Proof. We prove both directions.

(1. ⇒ 2.) Assume that $\mathcal{B}$ has a CPL$(\Lambda, \Sigma, \Gamma \cup \Gamma(A))$-expansion $\mathcal{B}^+ = (B, \beta, J^+)$ such that $\mathcal{B}^+ \models \mathrm{ElCoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A})$. It is to show that there is an elementary CPL$(\Lambda, \Sigma, \Gamma)$-embedding $h : A \to B$. Define h as in Lemma 4.14;

$$h : a \mapsto J^+(c_a) \text{ for all } a \in A. \tag{22}$$

By Lemma 4.14 we know that h is a CPL-embedding so it is left to show that h is elementary. Take any $a_1, .., a_n$ from A and any CPL$(\Lambda, \Sigma, \Gamma)$-formula $\varphi(x_1, .., x_n)$. We have

$$\begin{aligned}
\mathcal{A} \models \varphi(a_1, .., a_n) &\Leftrightarrow \varphi(c_{a_1}, .., c_{a_n}) \in \mathrm{ElCoDiag}(\mathcal{A}) && \text{(definition 4.13)} \\
&\Leftrightarrow \mathcal{B}^+ \models \varphi(c_{a_1}, .., c_{a_n}) && \text{(assumption)} \\
&\Leftrightarrow \mathcal{B} \models \varphi\big(h(a_1), .., h(a_n)\big) && \text{(by (22)).}
\end{aligned}$$

It is important to note that we are talking about CPL$(\Lambda, \Sigma, \Gamma)$-formulas *without* the constants c_a. So the last bi-implication is trivial, just as in the case for the predicate $P \in \Sigma$ in the proof for Lemma 4.14.

$(2. \Rightarrow 1.)$ Now assume that there is an elementary CPL-embedding $h : A \to B$. It is to show that $\mathcal{B}$ has a $\mathrm{CPL}(\Lambda, \Sigma, \Gamma \cup \Gamma(A))$-expansion $\mathcal{B}^+ = (B, \beta, J^+)$ such that $\mathcal{B}^+ \models \mathrm{ElCoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A})$. We will prove the stronger statement that for any sentence $\varphi \in \mathrm{CPL}(\Lambda, \Sigma, \Gamma)$ we have

$$\varphi \in \mathrm{ElCoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A}) \Leftrightarrow \mathcal{B} \models \varphi.$$

So take any sentence $\varphi(c_{a_1}, .., c_{a_n}) \in \mathrm{CPL}(\Lambda, \Sigma, \Gamma)$. We have

$$\begin{aligned}
\varphi(c_{a_1}, .., c_{a_n}) \in \mathrm{ElCoDiag}_{(\Lambda,\Sigma)}(\mathcal{A}) &\Leftrightarrow \mathcal{A} \models \varphi(c_{a_1}, .., c_{a_n}) \\
&\Leftrightarrow \mathcal{A} \models \varphi(a_1, .., a_n) \\
&\Leftrightarrow \mathcal{B} \models \varphi(h(a_1), .., h(a_n)) \\
&\Leftrightarrow \mathcal{B} \models \varphi(c_{a_1}, .., c_{a_n}).
\end{aligned}$$

We conclude that $\mathcal{B}^+ \models \mathrm{ElCoDiag}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{A})$. □

It is very important to stress that the coalgebraic diagram actually has nothing coalgebraic to its definition. It is the diagram of *coalgebraic* models but - as you can see in the proof - we never use anything else from the definitions of coalgebras. The name thus refers to CPL as a language with models that are coalgebras and not to anything else coalgebraically. In Chapter 6 we will put the Coalgebraic Diagram to use extensively.

Remark In the beginning of this chapter, we motivated our choice of not including any requirements on the coalgebra structure of elementary substructures in CPL by quoting Litak et al. [5]. Leaving coalgebra structure out of definition 4.1 is something once can also learn by trying different definitions and trying if the CoDiagram still characterises elementary CPL-embeddings as desired. If one for example adds to definition 4.1 that a strong CPL-homomorphism should be a coalgebra morphism, only the first direction of theorem 4.15 can be shown, as proposition 4.3 showcases.

On a related note: in the process of finding the final statement of theorem 4.15, which depends on the definition of the CoDiagram as well as the definition of a CPL-embedding, we have tried to enrich the CPL-language with predicates corresponding to all subsets of the carrier set of the model for which the CoDiagram is defined. We attempted to capture these extra predicates by including the Chang-condition in definition 4.1. However, in this approach, only the *second* direction of the theorem would work since it was not possible to argue about the validity of formulas in the enriched language. Through attempts of assuming e.g separation of the set of predicate liftings to say something about the CoDiagram, we realised that CPL-morphisms should be defined as we have done in section 4.1.

5 An Ehrenfeucht-Fraïssé-game for CPL

This chapter introduces a version of Ehrenfeucht-Fraïssé games (EF-games) for CPL, restricted to the case where Λ consists of unary and monotone predicate liftings only. We expect the reader to be familiar with classical EF-games for FOL[16] and it is good to think of the game described in this chapter of an alteration of the classical game to accommodate for coalgebra structure and predicate liftings. The main idea is that, just like existential formulas, modal formulas need to be *unpacked* by a particular move in the game. In section 5.1 we define the game and its winning conditions, in section 5.2 we prove the adequacy of the game and in section 5.3 we go through some concrete examples. Finally, we make some comments about the inclusion of non-unary predicate liftings in section 5.4.

5.1 The Game

An EF-game is played by two players between two models to examine the equivalence between the two models. The players are usually referred to as *Spoiler* and *Duplicator*, who we also refer to as she and he respectively, due to the binary nature of the game. Spoiler intends to show that the models are different while Duplicator intends to show that they are similar. Before the start of the game the number of rounds that will be played is determined and throughout the game Spoiler starts each round by performing a move in a model of choice. Duplicator then reacts by playing the same kind of move but in the other model. Doing so, the players construct two sequences in the two models and depending on the properties of those sequences, one of the players wins. It must be noted here that any finite game of the kind that we describe below always has a player with a winning strategy[17]. We call a specific instance of a game, with specific moves made by the players, a *match*.

For sections 5.1 and 5.2, fix a *finite* CPL-language $(\Lambda, \Sigma, \Gamma)$.

Assumption 5.1. *All predicate liftings $\heartsuit \in \Lambda$ are unary and monotone.*

Notationally convenient due to the binary nature of EF-games, we will usually play games between CPL-models $\mathcal{A}_0$ and $\mathcal{A}_1$. As the syntax of CPL includes the first-order quantifiers, we include the classical $\exists$-move.

[16] See for example [9, p. 52] or [7, p. 27].

[17] This is based on Zermelo's theorem.

Step 1a	Spoiler picks a model $\mathcal{A}_i$ for $i \in \{0,1\}$
Step 1b	Spoiler picks an element $a \in A_i$
Step 2	Duplicator picks an element $b \in A_{1-i}$

Table 1: The $\exists$-move

The $\exists$-move - so to speak - unpacks existential formulas. We define another kind of move to unpack the predicate liftings, inspired by EF-games for generalised quantifiers[18]. Given a coalgebra and an element in its domain, we associate each $\heartsuit \in \Lambda$ with a set of subsets of the domain.

Definition 5.2. *Fix a T-coalgebra $\mathcal{A} = (A, \alpha)$, an element $a \in A$ and a predicate lifting $\heartsuit \in \Lambda$. We set*

$$\mathcal{A}^{\heartsuit}(a) := \{X \subseteq A \mid \alpha(a) \in [|\heartsuit|]_A(X)\}.$$

Note that $\mathcal{A}^{\heartsuit}(a)$ is independent of an interpretation I. A CPL-move is defined as follows (see figure 1 and table 2). Assume that the players have established a configuration of two sequences $\vec{a}_0 = (a_0^1, .., a_0^k)$; $\vec{a}_1 = (a_1^1, .., a_1^k)$ in the two structures, for $k \in \omega$. Spoiler chooses some $\heartsuit \in \Lambda$ and a model $\mathcal{A}_i$, for $i \in \{0,1\}$. She proceeds by choosing an element $a_i^j \in \vec{a}_i$ in one of the previously chosen sequences and she then chooses an element C from $\mathcal{A}_i^{\heartsuit}(a_i^j)$: a subset of A_i. After this, Duplicator chooses an element D of $\mathcal{A}_{1-i}^{\heartsuit}(a_{1-i}^j)$, where a_{1-i}^j is the element in A_{1-i} that forms a pair with a_i^j in the foregoing configuration. Spoiler continues by choosing some $d \in D$, after which Duplicator chooses some $c \in C$. If, without loss of generality, we assume that Spoiler picked from A_0 first, we now have the extended sequences $(a_0^1, .., a_0^k, c)$ and $(a_1^1, .., a_1^k, d)$, noticing that the subsets C and D were merely auxiliary. If Spoiler's choice of $\heartsuit$ is known we may also refer to a CPL-move as a $\heartsuit$-move. Furthermore, we refer to e.g. the choosing of $C \in \mathcal{A}_i^{\heartsuit}(a_i^j)$ by Spoiler as a $\heartsuit$-move *on* a_i^j.

[18] See for example [10]; each quantifier is associated with a collection of subsets of a model.

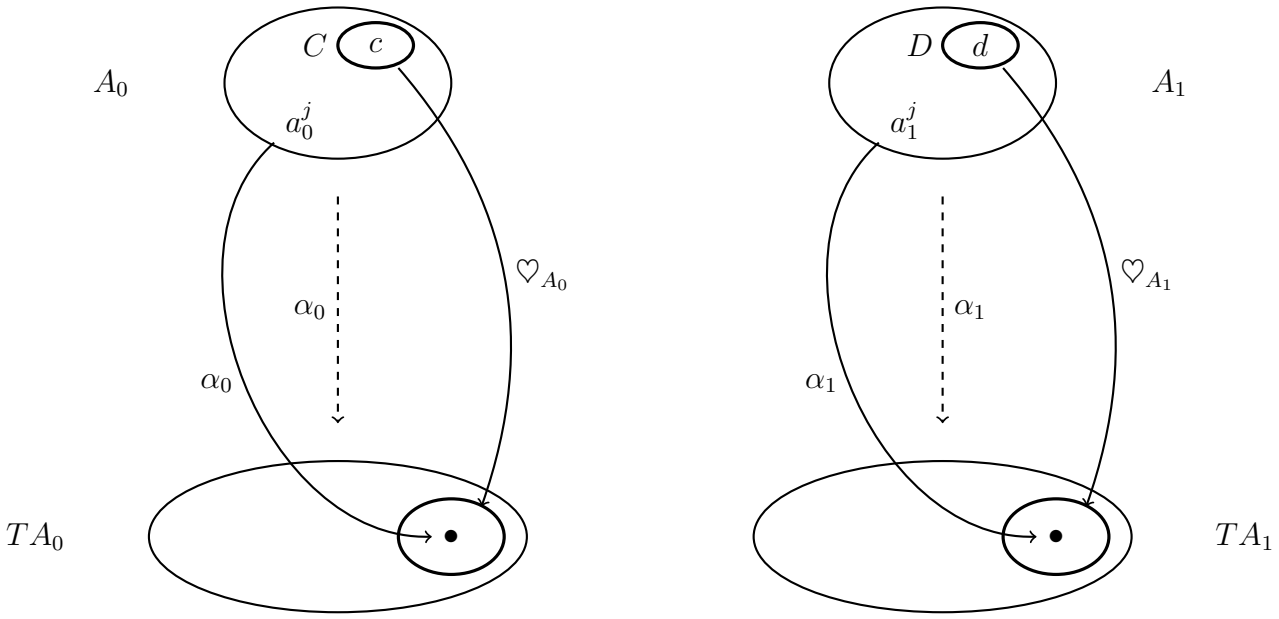


Figure 1: A $\heartsuit$-move between models $\mathcal{A}_0$ and $\mathcal{A}_1$.

Old Configuration:	$(\vec{a}_0; \vec{a}_1)$
Step 1	Spoiler picks some$\heartsuit \in \Lambda$
Step 2	Spoiler picks $\mathcal{A}_i$ for $i \in \{0, 1\}$
Step 3	Spoiler picks $a_i^j \in \vec{a_i}$
Step 4a	Spoiler picks $C \in \mathcal{A}_i^{\heartsuit}(a_i^j)$
Step 4b	Duplicator picks $D \in \mathcal{A}_{1-i}^{\heartsuit}(a_{1-i}^j)$
Step 5a	Spoiler picks $d \in D$
Step 5b	Duplicator picks $c \in C$
New configuration:	$(\vec{a}_0, c\ ;\ \vec{a}_1, d)$

Table 2: Overview of the $\heartsuit$-round

The EF-game for CPL consists of a fixed number of rounds and Spoiler decides the moves that are played each round. We denote a game of $n \in \omega$ rounds with predicate liftings Λ by $\mathrm{EF}_{\Lambda}^{n}(\mathcal{A}_0, \mathcal{A}_1)@(\vec{a}_0, \vec{a}_1)$, where $\vec{a}_0,\ \vec{a}_1$ are k-tuples, for $k \in \omega$, in the respective models, forming the starting configuration. We do allow the players to select an element more than once, as in one of the examples in section 5.3. The instance where $k = 0$ amounts

to a game with an empty starting configuration, forcing the first move to be an $\exists$-move. For the definition of the winning conditions, we need the notion of a local CPL-isomorphism.

Definition 5.3. *Fix $k \in \omega$ and a set of variables $\{x_1, .., x_k\} \subseteq iVar$. A local $CPL(\Lambda, \Sigma, \Gamma)-$isomorphism between two $CPL(\Lambda, \Sigma, \Gamma)$-models $\mathcal{A}_0 = (A_0, \alpha_0, I_0)$ and $\mathcal{A}_1 = (A_1, \alpha_1, I_1)$ is a function $f : C \subseteq_\omega A_0 \to A_1$, where $C = \{c_1, .., c_k\}$, such that*

$$\mathcal{A}_0 \models \varphi(c_1, .., c_k) \Leftrightarrow \mathcal{A}_1 \models \varphi\big(f(c_1), .., f(c_k)\big)$$

for all atomic $CPL(\Lambda, \Sigma, \Gamma)$-formulas $\varphi(x_1, .., x_k)$.

Duplicator wins the match $\mathrm{EF}^n_\Lambda(\mathcal{A}_0, \mathcal{A}_1)@(\vec{a}_0, \vec{a}_1)$ with plays $a_0^{k+1}, .., a_0^{k+n}$ and $a_1^{k+1}, .., a_1^{k+n}$ if and only if the partial function $f : A_0 \to A_1$ defined by

$$f(a_0^j) = a_1^j \text{ for } 0 \leq j \leq (k+n)$$

is a local CPL-isomorphism. Furthermore, Duplicator can only win if he always has a possible move, as is made explicit in one of the examples below. Clearly, Spoiler wins if and only if Duplicator does not.

5.2 Adequacy of the Game

To be able to use the game to actually assess how similar models are, we show that Duplicator winning the game is equivalent to the models making true the same formulas of a certain depth. Before we can formulate the theorem that describes this adequacy of the game, we need to specify with some more auxiliary definitions.

Definition 5.4. *(a) The combined depth $dp(\varphi(\vec{x}))$ of a formula $\varphi(\vec{x}) \in CPL(\Lambda, \Sigma, \Gamma)$ is defined inductively on the complexity of a formula, where s, t and $\vec{s}$ are terms and $P \in \Sigma$:*

$$\begin{aligned} dp(s = t) = dp(P(\vec{s})) = dp(\bot) &= 0 \\ dp(\varphi \to \psi) &= max(dp(\varphi), dp(\psi)) \\ dp(\exists x\varphi) = dp(\forall x\varphi) &= dp(\varphi) + 1 \\ dp(x \heartsuit \lceil y : \varphi \rceil) &= dp(\varphi) + 1 \end{aligned}$$

(b) For any $n \in \omega$, let $CPL_n(\Lambda, \Sigma, \Gamma)$ be all formulas of combined depth at most n:

$$\varphi \in CPL_n(\Lambda, \Sigma, \Gamma) \Leftrightarrow dp(\varphi) \leq n$$

(c) Let $\mathcal{A}_0$ and $\mathcal{A}_1$ be two $CPL(\Lambda, \Sigma, \Gamma)$-models. Let $\vec{a}_0$ and $\vec{a}_1$ be two sequences of length k in A_0 and A_1 respectively. Then we write $\mathcal{A}_0, \vec{a}_0 \equiv_n \mathcal{A}_1, \vec{a}_1$ iff

$$\mathcal{A}_0 \models \varphi(\vec{a}_0) \Leftrightarrow \mathcal{A}_1 \models \varphi(\vec{a_1})$$

for all $\varphi(x_1, .., x_k)$ in $CPL_n(\Lambda, \Sigma, \Gamma)$.

Alternatively, one could count quantifier-depth and modal depth separately. We will say a little more about that in the last section of this chapter.

For the proof of the adequacy theorem it is of key importance that the number of CPL-formulas of a certain combined depth is finite up to logical equivalence. Recall that we are working under the assumption that our CPL-language is finite.

Proposition 5.5. *Let $n, k \in \omega$. Then the number of formulas in $CPL_n(\Lambda, \Sigma, \Gamma)$ with free variables among $\{x_1, .., x_k\}$ is finite up to logical equivalence.*

Proof. This can be proven by induction on n. For the base case $n = 0$ notice that there are only finitely many predicates and one logical constant $\bot$ and k-many variables. So there are finitely many atomic formulas. The Boolean closure of the set of all atomic formulas, which all have combined depth 0, is then finite up to logical equivalence.

Assume for the inductive step that, up to logical equivalence, there are finitely many formulas with free variables among $\{x_1, .., x_k\}$ of depth $n - 1$. Then, for each $\varphi \in CPL_{n-1}(\Lambda, \Sigma, \Gamma)$, there are only finitely many ways to create a formula of depth n by using one of $\heartsuit \in \Lambda$ or $\exists$. Picking only one formula for each class of logically equivalent formulas and noticing again that the Boolean closure of the resulting set of CPL_n-formulas is then also finite up to logical equivalence, we conclude that effectively there are finitely many formulas of combined depth n with free variables among $\{x_1, .., x_k\}$. □

We are now ready to describe and prove the adequacy of the EF-game for CPL.

Theorem 5.6 (Adequacy)**.** *Let $\mathcal{A}_0 = (A_0, \alpha_0, I_0)$ and $\mathcal{A}_1 = (A_1, \alpha_1, I_1)$ be two $CPL(\Lambda, \Sigma, \Gamma)$-models, let $k, n \in \omega$ and let $\vec{a}_0, \vec{a}_1$ be two k-sequences in A_0 and A_1 respectively. Then the following are equivalent:*

(i) $\mathcal{A}_0, \vec{a}_0 \equiv_n \mathcal{A}_1, \vec{a}_1$

(ii) Duplicator has a winning strategy in $EF^n_\Lambda(\mathcal{A}_0, \mathcal{A}_1)@(\vec{a}_0, \vec{a}_1)$

Proof. We prove the equivalence of (i) and (ii) by induction on n.

Base Case If $n = 0$ then (i) just says that $\mathcal{A}_0, \vec{a}_0$ and $\mathcal{A}_1, \vec{a}_1$ make true the same atomic formulas in k free variables. This coincides with a winning position for Duplicator after 0 rounds. So the equivalence follows trivially from the definitions.

Inductive step For the inductive case, we treat both directions separately.

From (i) to (ii) Assume $\mathcal{A}_0, \vec{a}_0 \equiv_n \mathcal{A}_1, \vec{a}_1$. We play the game $\mathrm{EF}^n_\Lambda(\mathcal{A}_0, \mathcal{A}_1)@(\vec{a}_0, \vec{a}_1)$ and describe how Duplicator can react to any move by Spoiler in such a way that we can use the induction hypothesis for the game with $n-1$ rounds. There are two cases: Spoiler can either (1) do a $\exists$-move or (2) do a $\heartsuit$-move.

Case (1) Spoiler does a $\exists$-move. Consider all formulas in $\mathrm{CPL}_{n-1}(\Lambda, \Sigma, \Gamma)$ in $k+1$ free variables. The first k variables $\vec{x}$ will be assigned accordingly to the current configuration and the last variable y is for the pair that will be chosen in the current move; we will bind y with an existential quantifier. Let Spoiler pick some $c \in A_0$. We define the formula:

$$\varphi_\exists(\vec{x}, y) \quad := \quad \bigwedge\{\psi(\vec{x}, y) \quad \in \quad \mathrm{CPL}_{n-1}(\Lambda, \Sigma, \Gamma) \quad | \quad \mathcal{A}_0 \quad \models \quad \psi(\vec{a}_0, c)\} \tag{23}$$

Here $\neg\{\psi_1, ..\psi_n\}$ should be understood as $\{\neg\psi_1, .., \neg\psi_n\}$. Notice that $\varphi_\exists$ might in general be an infinite conjunction. However, by Proposition 5.5 we may pick representatives for each equivalence class under logical equivalence, yielding finitely many conjuncts and thus a well-defined (finite) formula $\varphi_\exists$. By construction we have $\mathcal{A}_0 \models \varphi_\exists(\vec{a}_0, c)$. So $\mathcal{A}_0 \models \exists y \varphi_\exists(\vec{a}_0, y)$. We have $\mathcal{A}_0, \vec{a}_0 \equiv_n \mathcal{A}_1, \vec{a}_1$ which means that $\mathcal{A}_1 \models \exists y \varphi_\exists(\vec{a}_1, y)$. This gives us an element $d \in A_1$ such that $\mathcal{A}_1 \models \varphi_\exists(\vec{a}_1, d)$. It should be obvious that Duplicator picks d. By construction of $\varphi_\exists$, we have $\mathcal{A}_0, \vec{a}_0, c \equiv_{n-1} \mathcal{A}_1, \vec{a}_1, d$.

Case (2) Spoiler does a CPL-move; without loss of generality she picks $\mathcal{A}_0$, some $\heartsuit \in \Lambda$ and some $C \in \mathcal{A}_0^\heartsuit(a_0^j)$ for some a_0^j among $\vec{a}_0$. We again consider all $\psi \in \mathrm{CPL}_{n-1}(\Lambda, \Sigma, \Gamma)$ with $k+1$ free variables. Again k variables for the already chosen sequences and now a last variable y that is bound in the formula that $\heartsuit$ is applied to. We now define:

$$\varphi_\heartsuit(\vec{x}, y) \quad := \quad \bigwedge\{\psi(\vec{x}, y) \quad \in \quad \mathrm{CPL}_{n-1}(\Lambda, \Sigma, \Gamma) \quad | \quad C \quad \subseteq \quad [|\psi(\vec{x}, y)|]^{\vec{a}_0, y}_{\mathcal{A}_0}\} \tag{24}$$

For the same reasons as above, $\varphi_\heartsuit$ may be assumed to be finite. By construction we have that if $a \in C$ then $a \in [|\varphi_\heartsuit(\vec{x}, y)|]^{\vec{a}_0, y}$. By definition of $\mathcal{A}_0^\heartsuit(a_0^j)$, we know that $\alpha_0(a_0^j) \in [|\heartsuit|]_{A_0}(C)$ and so we have

$$\alpha_0(a_0^j) \in [|\heartsuit|]_{A_0}(C) \subseteq [|\heartsuit|]_{A_0}([|\varphi_\heartsuit|]^{\vec{a}_0, y}_{\mathcal{A}_0})$$

by monotonicity. So $\mathcal{A}_0 \models a_0^j \heartsuit \lceil y : \varphi_\heartsuit(\vec{a}_0, y)\rceil$. Now notice that

$$dp(\varphi_\heartsuit) = max\big(dp(\psi)_{\psi \in \mathrm{CPL}_{n-1}(\Lambda, \Sigma, \Gamma)}\big) = n - 1 \tag{25}$$

and $dp(x \heartsuit \lceil y : \varphi_\heartsuit\rceil) = n - 1 + 1 = n$. Together with $\mathcal{A}_0, \vec{a}_0 \equiv_n \mathcal{A}_1, \vec{a}_1$ we have $\mathcal{A}_1 \models a_1^j \heartsuit \lceil y : \varphi_\heartsuit(\vec{a}_1, y)\rceil$. So we let Duplicator pick $D := [|\varphi_\heartsuit(\vec{x}, y)|]^{\vec{a}_1, y}_{\mathcal{A}_1}$. Then we have $\alpha_1(a_1^j) \in [|\heartsuit|]_{A_1}(D)$ so $D \in \mathcal{A}_1^\heartsuit(a_1^j)$, justifying the reaction made by Duplicator as a valid move.

Next, Spoiler picks some $d \in D \subseteq A_1$ to which Duplicator may react with any $c \in C \subseteq A_0$. Notice that $d \in [|\varphi_\heartsuit(\vec{x}, y)|]^{\vec{a}_0, y}_{\mathcal{A}_0}$ and $c \in [|\varphi_\heartsuit(\vec{x}, y)|]^{\vec{a}_1, y}_{\mathcal{A}_1}$. By the construction of $\varphi_\heartsuit$, c and d make true exactly the same formulas of depth $n-1$ in $k+1$ variables. In other words, for all formulas ψ in $(k+1)$-many variables and of depth $n-1$, we have $\mathcal{A}_0 \models \psi(\vec{a}_0, c) \Leftrightarrow \mathcal{A}_1 \models \psi(\vec{a}_1, d)$ so by definition we have $\mathcal{A}_0, \vec{a}_0, c \equiv_{n-1} \mathcal{A}_1, \vec{a}_1, d$.

In both cases, the winning strategy is formed by playing the described reaction to any play by Spoiler and then appending the play with the winning strategy for $\mathrm{EF}^{n-1}_\Lambda(\mathcal{A}_0, \mathcal{A}_1)@(\vec{a}_0, c; \vec{a}_1, d)$ given by the induction hypothesis.

From (ii) to (i) In the other direction, we go by contraposition. We assume $(\star)$ $\mathcal{A}_0, \vec{a}_0 \not\equiv_n \mathcal{A}_1, \vec{a}_0$ and show a winning strategy for Spoiler. By $(\star)$, there is a formula $\varphi(x_1, .., x_k)$ of depth n such that $\mathcal{A}_0 \models \varphi(\vec{a}_0)$ but $\mathcal{A}_1 \not\models \varphi(\vec{a}_1)$. There are two possibilities for the main operator of $\varphi(x_1, .., x_k)$.

Poss. (1) If $\varphi(x_1, .., x_k)$ is of the form $\exists x \psi(x, x_1, .., x_k)$, we know two things. First, there is some assignment of $v_0 : iVar \to A_0$ such that $x \mapsto c$ for some $c \in A_0$ and $\mathcal{A}_0, v_0 \models \psi(a_0, \vec{a}_0)$. Second, there is no assignment $v_1 : iVar \to A_1$ such that $\mathcal{A}_1, v_1 \models \psi(v_1(x), \vec{a}_1,)$. This means Spoiler can pick $c \in A_0$ by doing an $\exists$-move and that no matter what element $d \in A_1$ Duplicator replies with, we always have $\mathcal{A}_1 \not\models \psi(c, \vec{a}_1)$. So then $\mathcal{A}_0, \vec{a}_0, c \not\equiv_{n-1} \mathcal{A}_1, \vec{a}_1, d$.

Poss. (2) If $\varphi(x_1, .., x_k)$ is, without loss of generality, of the form $x_1 \heartsuit \lceil y : \psi(x_2, .., x_k, y) \rceil$ we have

$$\mathcal{A}_0 \models a^1_0 \heartsuit \lceil y : \psi(a^2_0, .., a^k_0, y) \rceil \text{ and } \mathcal{A}_1 \not\models a^1_1 \heartsuit \lceil y : \psi(a^2_1, .., a^k_1, y) \rceil,$$

which means that

$$\alpha_1(a^1_0) \in [|\heartsuit|]_{A_0}([|\psi(a^2_0, .., a^k_0, y)|]^y_{A_0}) \text{ but } \alpha_1(a^1_1) \notin [|\heartsuit|]_{A_1}([|\psi(a^2_1, .., a^k_1, y)|]^y_{A_1}).$$

Let Spoiler pick $[|\psi(a^2_0, .., a^k_0, y)|]^{,y}_{\mathcal{A}_0} \in \mathcal{A}^\heartsuit_0(a^1_0)$. Duplicator now has to reply with some $D \in \mathcal{A}^\heartsuit_1(a^1_1)$. If he picks a D such that $D \subseteq [|\psi(a^2_1, .., a^k_1, y)|]^y_{\mathcal{A}_1}$ then we would have

$$\alpha_1(a^1_1) \in [|\heartsuit|]_{A_1}(D) \subseteq [|\heartsuit|]_{A_1}([|\psi(a^2_1, .., a^k_1, y)|]^y_{\mathcal{A}_1}),$$

by monotonicity. But we just saw that $\alpha_1(a^1_1) \notin [|\heartsuit|]_{A_1}([|\psi(a^2_1, .., a^k_1, y)|]^{,y}_{\mathcal{A}_1})$ so we must have $D \not\subseteq [|\psi(a^2_1, .., a^k_1)|]^y_{\mathcal{A}_1}$. Then there is a $d \in D$ such that $d \notin [|\psi(a^2_1, .., a^k_1)|]_{\mathcal{A}_1}$. This means that Spoiler should now pick this element $d \in D$. On the other model, Duplicator has no choice but to pick some $c \in [|\psi(a^2_0, .., a^k_0)|]^y_{\mathcal{A}_0}$. So we get $\mathcal{A}_0, \vec{a}_0, a \not\equiv_{n-1} \mathcal{A}_1, \vec{a}_1, d$.

In both cases we have $\mathcal{A}_0, \vec{a}_0, c \not\equiv_{n-1} \mathcal{A}_1, \vec{a}_1, d$ so we can use the inductive hypothesis to extend Spoiler's winning strategy.

Since we covered both directions of the inductive step, the induction is complete and we conclude that the EF-game for CPL is adequate. □

Note that the cases in the proof where Spoiler uses the $\exists$-move are the same as in the classical EF-game for FOL.

5.3 Examples

Now that we have established the game and its adequacy, we turn to our three running examples. The two exemplary games below both have empty starting configurations.

Relational FOL Recall that $T = \mathcal{P}$[19] and $\Lambda = \{\Diamond\}$. For simplicity, take $\Sigma = \Gamma = \emptyset$ so that we don't have to specify interpretations. For a $\Diamond$-move we follow definition 5.2. Let $\mathcal{A} = (A, \alpha)$ be a $\mathcal{P}$-model and let $a \in A$. Then

$$\mathcal{A}^{\Diamond}(a) := \{X \subseteq A \mid \alpha(a) \in [|\Diamond|]_A(X)\} = \{X \subseteq A \mid X \cap \alpha(a) \neq \emptyset\}.$$

So $\mathcal{A}^{\Diamond}(a)$ is the collection of all $X \subseteq A$ that contain at least one **successor** of a. If a player picks such an X, it makes sense to always pick a singleton, effectively forcing their opponent to pick a certain successor. So the $\Diamond$-move looks a lot like the move in a bisimulation game for ordinary modal logic[20].

To illustrate how the game works and how it is different from the classical EF-game, let us look at a concrete example, where we depict the models as Kripke frames.

To show that only $\mathcal{A}_0$ has an element that 'sees' two distinct elements, Spoiler may notice that the models do not agree on the following formula:

$$\exists x \exists y \exists w (x \Diamond \lceil z : z = y \rceil \wedge y \neq w \wedge x \Diamond \lceil z : z = w \rceil)$$

The formula has a combined depth of 4 so Spoiler should be able to win in as many rounds. The players may play the following match, where one should notice that Duplicator has not much choice. Recall that we assumed there are no first-order predicates or constants, so the only atomic formulae are of the form $x = y$.

[19] For simplicity, we assume there are no nullary predicates.

[20] See for example [11, p. 257].

Moves 1-3 First three $\exists$-rounds where Spoiler and Duplicator select the colour-pairs as indicated in the figure (e.g.: in round 1, Spoiler picks the green element in $\mathcal{A}_0$ and Duplicator the green element in $\mathcal{A}_1$).

Picking the elements can be seen as giving names to the elements. In the order green-blue-red, we have: a_1, a_2, a_3 in $\mathcal{A}_0$ and b_1, b_2, b_3 in $\mathcal{A}_1$ (e.g.: blue in $\mathcal{A}_1$ is called b_2).

Move 4 See the thicker arrows in the figure. Spoiler does a $\Diamond$-move on a_1 in $\mathcal{A}_0$ (green) and picks the element that is not paired up with a successor of b_1 in $\mathcal{A}_1$, namely a_3. With this move, we give red in $\mathcal{A}_0$ another name: c.

Duplicator has to reply by picking a set D that includes the only successor of b_1: b_2. Spoiler then picks $b_2 \in D$, this time calling it d.

It should now be clear that Duplicator has lost the game since the elements that the players picked with the $\Diamond$-move to do not form a pair in the configuration after the first three moves. Formally, the names given to the elements in the two models do not match. We have

$$\mathcal{A}_0 \models a_3 = c \text{ and}$$
$$\mathcal{A}_1 \not\models b_3 = d.$$

Keep in mind that the map defined by the game sends $f(a_3) = b_3$ and $f(c) = d$ so f is not a local isomorphism.

The most important take-away of the example of the EF-game for Relational FOL described by CPL is that the the relation defined by the coalgebra map is incorporated into the game. The point is that local isomorphisms only look at *atomic* formulas. In CPL, modality formulas are not atomic so we need an extra move to unpack them. In other words, the fact that formulas like $x\Diamond\lceil y : y = z\rceil$ are not atomic, forces us to do two things, on both sides of the adequacy theorem: (1) we need to view it as a formula with depth 1 and (2) we need to make sure there is a way to unpack the depth to the subformula $y = z$ by adding the $\Diamond$-move.

In a classical version, the atomic formula which would make the partial isomorphism fail is some relation R that describes the relation defined by the arrows in the figure above. So the game would only last 3 rounds. This is because the translation of a formula of the form $x\Diamond\lceil y : y = z\rceil$ only involves some relation R (and no modal operators or quantifiers), which makes it an atomic formula.

Neighbourhood Logic Recall that we have $T = \mathcal{Q}\mathcal{Q}$ and $\Lambda = \{\Box\}$. For a $\Box$-move on a model $\mathcal{A}$, we look at the following set:

$$\mathcal{A}^{\Box}(a) = \{X \subseteq A \mid \alpha(a) \in [|\Box|]_A(X)\} = \{X \mid X \in \alpha(a)\ \} = \alpha(a)$$

So in the game the players **pick a neighbourhood** of a. The second step of the $\Box$-move is picking an element in the neighbourhood that your opponent picked.

Note that the neighbourhood modality for $\mathcal{QQ}$ is not usually assumed to be monotone. However, we need monotonicity for the game to be worked.

Graded Modal Logics Recall that $T = \mathcal{B}$ and the set of predicate liftings $\{\langle k\rangle \mid k \geq 0\}$. So there is a $\langle k\rangle$-move for every $k \geq 0$. Given a $\mathcal{B}$-model $\mathcal{A}$, we have

$$\mathcal{A}^{\langle k\rangle}(a) = \{X \subseteq A \mid \alpha(s) \in [|\langle k\rangle|]_A(X)\} = \{X \subseteq A \mid \alpha(a)[X] > k\}$$

So players choose from the collection of sub-domains on which $\alpha(a)$ has an outcome bigger than k.

Now consider the model $\mathcal{A}_0$ with $A = \{a, b\}$ and the model $\mathcal{A}_1$ with $A_1 = \{c, d\}$ with coalgebraic structures defined in the following intuitive picture (which does not depict it coalgebraically). To read the picture, we e.g. have $\alpha(a)(b) = 3$.

If we were to play a game with these models, Spoiler may notice that $\mathcal{A}_1$ has an element with total 'successor-weight' more than 6 (element d), while $\mathcal{A}_0$ does not. Consider the formula

$$\exists x x\langle 6\rangle\lceil y : y = y\rceil. \tag{26}$$

We use the tautology $y = y$ just to get the whole domain of the model, so to get the total 'successor-weight' of x, which should then be higher than 6. Now the 2-round game goes as follows:

Move 1 An $\exists$-move. Spoiler picks $d \in A_1$, Duplicator reacts, without loss of generality, with $a \in S_0$

Move 2 A $\langle 6\rangle$-move. Spoiler picks $\{c, d\} \in \mathcal{A}_1^{\langle 6\rangle}(d)$. Duplicator has to pick some $X \in \mathcal{A}_0^{\langle 6\rangle}(a)$ but there is no possibility since $\alpha(a)[A_0] = 6$.

Duplicator automatically loses the game since there is no possible move. We can see that in general, if Spoiler wants to show that $x\langle k\rangle\lceil y : \varphi\rceil$ is true in $\mathcal{A}_i$ and not in $\mathcal{A}_{1-i}$, she may pick $[|\varphi|] \subseteq A_i$.

Remark 5.7. *For an arbitary set-functor T it is a bit more difficult to determine how the game will be played but it makes sense for e.g. Spoiler to pick $[|\varphi|]$ in one of the models if they suspect that a formula of the form $x\heartsuit\lceil y : \varphi\rceil$ is only true in that model, as it made explicit in the adequacy proof above.*

5.4 About Non-unary Modalities

In the above, we assumed all the predicate liftings in our similarity type to be unary. Of course, one could also generalise CPL-moves to liftings with arbitrary finite arity. The game itself as well as the proof would then become a bit more complex. We take a brief look at how to generalise definition 5.2 and how that would change the game.

Let us first consider nullary predicate liftings. The set $\mathcal{A}^{\heartsuit}(a)$ is based on the fact that $\heartsuit_A$ takes subsets of A as arguments but in the case that $\heartsuit$ is nullary, each subset of A would give the same subset. So a CPL-move for nullary predicate liftings would just give the players a chance to first each select a subset of the respective domains and then an element out of the sets that the other picked. Players would probably not use this, since they can also just employ the $\exists$-move.

Now for the general case, with arity $n \geq 2$, the first change is that in definition 5.2 we would get a collection of n-tuples of subsets, for an n-ary modality.

Definition 5.8. *Fix a T-coalgebra $\mathcal{A} = (A, \alpha)$, an element $a \in A$ and an n-ary predicate lifting $\heartsuit \in \Lambda$, for $n \in \omega$. We set*

$$\mathcal{A}^{\heartsuit}(a) := \{(X_1, .., X_n) \mid X_i \subseteq A, \text{ for } i \leq n \text{ and } \alpha(a) \in [|\heartsuit|]_A(X_1, .., X_n)\}.$$

Extending this generalisation, a $\heartsuit$-move would consists of Spoiler and Duplicator first both picking tuples of subsets and consequently tuples of elements in each other's subsets. To account for this in the adequacy proof, one would have to generalise the statement of the adequacy theorem in way that respects the fact that each move might extend the configuration with more than 1 element. This change would also influence the number of free variables in the formulas in the statement, which would be dependent on what kind of moves are used throughout. This could of course get quite technical but is a good challenge for further research.

Something else that might be done differently is our approach towards the depth of CPL-formulas. We treated the quantifiers and modal operators the same way but it is of course possible to separate the two. This would likely give an adequacy statement that is also able to say something about the number of $\exists$-moves and CPL-moves used throughout a match, instead of just the total number of moves. This is also something we leave open for further research.

In the next chapter, we start really implementing the definitions and results of chapter 4.

6 Directed Systems & Semantic Interpolation in CPL

In this chapter we look at an approach towards proving the Craig Interpolation Property (CIP) for certain versions of CPL via model theoretic methods. We do not state that we prove it because the construction includes a rather big assumption which might make one see this chapter as a description of why the approach does *not* work. In section 6.1 we introduce directed systems of CPL-models and the construction of their colimit. In section 6.1.1 we describe a major problem with this construction by giving a necessary assumption, which we work with in the rest of the chapter: in section 6.2 we prove some build-up lemmas and in section 6.3 we use the construction of a directed system to prove Robinson's consistency theorem for CPL, with which we show the interpolation theorem. The proofs of both Robinson's consistency and the CIP rely in multiple occasions on compactness, so we are only looking at specific instances of CPL.

6.1 Directed Systems

For the rest of this section, fix a set-functor T and a CPL-language $(\Lambda, \Sigma, \Gamma)$.

Let $(\mathcal{A}_k)_{k\in K}$ be a set of CPL$(\Lambda, \Sigma, \Gamma)$-models, indexed by a directed partial order K[21], where each $\mathcal{A}_k = (A_k, \alpha_k, I_k)$. Additionally, for any $k \leq l$, let there be a strong CPL$(\Lambda, \Sigma, \Gamma)$-morphism (recall definition 4.1) $f_{kl} : A_k \to A_l$ where:

- the f_{kk} are identities
- $f_{lm} \circ f_{kl} = f_{km}$ for all $m \geq l \geq k$

We call $(\mathcal{A}_k)_{k\in K}$, together with the strong CPL-morphisms f_{kl}, a directed system.

Given a directed system we want to define a colimit, together with strong CPL-morphisms from each model in the system to said colimit. The rest of this section describes how to do this and proves that it works. We first focus on the first-order part, defining the domain and first-order interpretation of the colimit and proving some basic propositions. After that, we will shift focus to the coalgebra structure of the colimit. The objective is then to show that there is a coalgebra map $\alpha : A \to TA$ that has certain properties. With the existence of such a coalgebra map we will then be able to prove the main theorem of this section, Theorem 6.15.

For the rest of the section, fix a directed system $(\mathcal{A}_k)_{k\in K}$ of CPL$(\Lambda, \Sigma, \Gamma)$-models, so that we can show that a colimit $\mathcal{A} = (A, \alpha, I)$ exists. Put $A = \bigsqcup_{k\in K} A_k/\!\sim$, where

$$(k, a) \sim (l, b) \Leftrightarrow \text{ there is some } m \geq k, l \text{ such that } f_{km}(a) = f_{lm}(b). \tag{27}$$

[21] That is, for all $k, l \in K$, there is some $m \in K$ such that $m \geq k, l$.

The carrier set A thus consists of equivalence classes $[k, a]$. Define maps $f_k : A_k \to A$ by putting

$$f_k : a \mapsto [k, a]. \tag{28}$$

Proposition 6.1. *For all $k, l \in K$, we have $f_l \circ f_{kl} = f_k$.*

Proof. Take any $k, l \in K$ and some $a \in A_k$. Then

$$f_l\big(f_{kl}(a)\big) = [l, f_{kl}(a)] \text{ and } f_k(a) = [k, a].$$

Furthermore, we have $l \geq l, k$ and $f_{ll}\big(f_{kl}(a)\big) = f_{kl}(a)$. So $[l, f_{kl}(a)] \sim [k, a]$, which implies $f_l \circ f_{kl} = f_k$. □

For the definition of I, take some $P \in \Sigma$, where P is $n-$ary, and let

$$\begin{aligned}([k_1, a_1], .., [k_n, a_n]) \in I(P) \Leftrightarrow \text{ there is some } l \geq k_1, .., k_n \\ \text{such that } \big(f_{k_1 l}(a_1), .., f_{k_n l}(a_n)\big) \in I_l(P).\end{aligned} \tag{29}$$

For each $c \in \Gamma$, we let

$$c^{\mathcal{A}} = [k_0, c^{\mathcal{A}_{k_0}}] \text{ for some } k_0 \in K. \tag{30}$$

By directedness of K it should be clear that I is well-defined.

Proposition 6.2. *All f_k are strong CPL$(\Lambda, \Sigma, \Gamma)$-morphisms.*

Proof. Take some $k \in K$. First, we have

$$(a_1, .., a_n) \in I_k(P) \Leftrightarrow \big(f_k(a_1), .., f_k(a_n)\big) \in I(P)$$

for all $a_1, .., a_n \in A_k$ and all $P \in \Sigma$. To see this, take arbitary $a_1, .., a_n \in A_k$ and some n-ary $P \in \Sigma$. For the first direction, we have

$$\begin{aligned}(a_1, .., a_n) \in I_k(P) &\Rightarrow \big(f_{kk}(a_1), .., f_{kk}(a_n)\big) \in I_k(P) && (f_{kk} \text{ is the identity}) \\ &\Rightarrow \big([k, a_1], .., [k, a_n]\big) \in I(P) && (k \geq k \text{ and definition (29)}) \\ &\Rightarrow \big(f_k(a_1), .., f_k(a_n)\big) \in I(P) && (\text{definition of } f_k)\end{aligned}$$

For the other direction, we have

$$\begin{aligned}\big(f_k(a_1), .., f_k(a_n)\big) &\Rightarrow \big([k, a_1], .., [k, a_n]\big) \in I(P) && (\text{definition of } f_k) \\ &\Rightarrow \big(f_{kl}(a_1), .., f_{kl}(a_n)\big) \in I_l(P) \text{ for some } l \geq k && (\text{definition (29)}) \\ &\Rightarrow (a_1, .., a_n) \in I_k(P) && (f_{kl} \text{ is a str. CPL-morph.})\end{aligned}$$

Second, we have $f_k(c^{\mathbf{A}_k}) = c^{\mathcal{A}}$ for all $c \in \Gamma$. Take arbitrary $c \in \Gamma$. Recall that $c^{\mathcal{A}} = [k_0, c^{\mathcal{A}_{k_0}}]$ for some $k_0 \in K$. By directedness there is some $k' \geq k_0, k$. We have

$$f_k(c^{\mathcal{A}_k}) = f_{k'}(f_{kk'}(c^{\mathcal{A}_k}) = f_{k'}(c^{\mathcal{A}_{k'}}) = [k', c^{\mathcal{A}_{k'}}]$$

and

$$c^{\mathcal{A}} = [k_0, c^{\mathcal{A}_{k_0}}] = [k', f_{k_0k'}(c^{\mathcal{A}_{k_0}})] = [k', c^{\mathcal{A}_{k'}}]$$

so $f_k(c^{\mathbf{A}_k}) = c^{\mathcal{A}}$. □

Proposition 6.3. *If all f_{kl} are injective, then so are all f_k.*

Proof. Take an arbitrary $k \in K$ and arbitary $a, b \in A_k$. Assume $a \neq b$. It is to show that $f_k(a) \neq f_k(b)$, i.e. $[k, a] \not\sim [k, b]$. Assume towards a contradiction that $[k, a] \sim [k, b]$. Then there is some $l \geq k$ such that $f_{kl}(a) = f_{kl}(b)$. By the assumed injectivity of f_{kl}, we get $a = b$. Contradiction so $[k, a] \not\sim [k, b]$; f_k is injective. □

Note that the above two Propositions in no way depend on the coalgebra map $\alpha : A \to TA$. So $\mathcal{A}$ need not be fully known to know that all f_k are CPL-embeddings.

Corollary 6.4. *If all f_{kl} are CPL$(\Lambda, \Sigma, \Gamma)$-embeddings, then so are all f_k.*

This concludes the first-order part of the colimit. Before we turn to showing the existence of a *suitable* coalgebra map $\alpha : A \to TA$, we want to take a short detour to appreciate how natural taking the colimit of a directed system of coalgebras is if all f_{kl} are coalgebra morphisms. Consider A as a category theoretical colimit in SET of the category $(A_k)_{k \in K}$; we can show the universality of the colimit.

Proposition 6.5. *For any set B with functions $g_k : A_k \to B$ such that $g_l \circ f_{kl} = g_k$ for every $k, l \in K$ with $k \leq l$, there is a unique map $g : A \to B$ such that $g \circ f_k = g_k$ for every $k \in K$.*

Proof. We explicitly define $g : A \to B$. Take some $[k, a] \in A$. Set

$$g : [k, a] \mapsto g_k(a). \tag{31}$$

Let us briefly check that g is well defined. Take $[k, a] \in A$ and assume $(l, b) \sim (k, a)$. Then there is some $m \geq k, l$ such that $f_{km}(a) = f_{lm}(b)$. By the assumption on the commutation of g_k with the f_{kl} we thus have

$$g_k(a) = g_m(f_{kl}(a)) = g_m(f_{lm}(b)) = g_l(b).$$

To see that (31) works a definition, take any $a \in A_k$. We have $g\big(f_k(a)\big) = g([k, a]) = g_k(a)$, as desired. For uniqueness of g, assume there is a $g' : A \to B$ such that $g' \circ f_k = g_k$ for all $k \in K$. If $g' \neq g$ then there is some $[k, a] \in A$ such that $g'([k, a]) \neq g([k, a])$. Now take $a \in A_k$ itself. We assumed $g'\big(f_k(a)\big) = g_k(a)$. But $g'\big(f_k(a)\big) = g'([k, a]) = g_k(a) = g([k, a])$. Contradiction so we must have $g' = g$. □

It is not hard to show that if all the $g_k : A_k \to B$ in Proposition 6.5 are $\mathrm{CPL}(\Lambda, \Sigma, \Gamma)$-embeddings, then so is the map $g : A \to B$.

If we assume all f_{kl} to be coalgebra morphisms, we get the coalgebra map $\alpha : A \to TA$ *for free* by universality. Continuing in categorical terms, we know that TA is the colimit of $(TA_k)_{k \in K}$, with maps $Tf_k : TA_k \to TA$. This is because the category Sets is co-complete and any set-functor preserves colimits.

Proposition 6.6. *If all f_{kl} are coalgebra morphisms, then so are all f_k.*

Proof. We start by using Proposition 6.5. In the case of $B = TA$, we have maps

$$Tf_k \circ \alpha_k : A_k \to TS_k \tag{32}$$

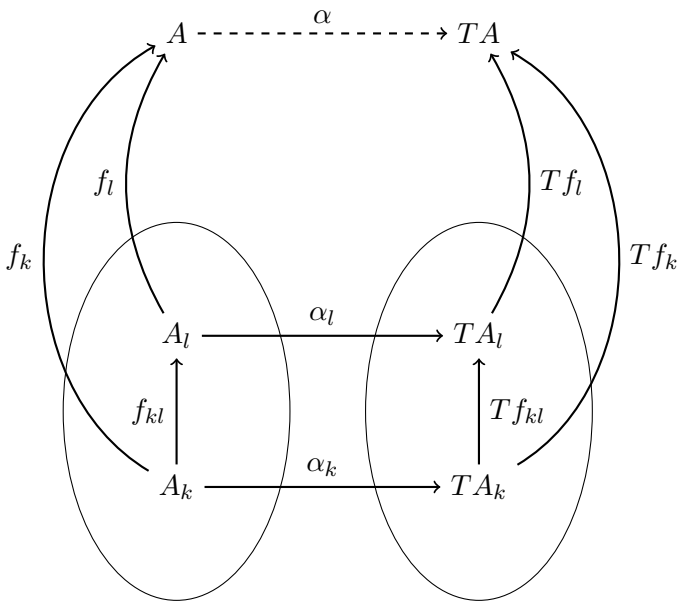
Furthermore, we also have

$$(Tf_l \circ \alpha_l) \circ f_{kl} = Tf_k \circ \alpha_k. \tag{33}$$

To see this, we employ the assumption that all f_{kl} are coalgebra morphisms. We have

$$\begin{aligned} Tf_l \circ \alpha_l \circ f_{kl} &= Tf_l \circ Tf_{kl} \circ \alpha_k \\ &= T(f_l \circ f_{kl}) \circ \alpha_k \\ &= Tf_k \circ \alpha_k \end{aligned}$$

By Proposition 6.5 there is a unique map $\alpha : A \to TA$ such that $\alpha \circ f_k = Tf_k \circ \alpha_k$ for every $k \in K$. This is precisely the condition for f_k to be a coalgebra morphism. □



However trivial the above construction of α, we cannot assume that all f_{kl} are in fact coalgebra morphisms. Without that assumption, showing that there is a coalgebra map $\alpha : A \to TA$ that makes all f_k elementary CPL-embeddings is based on the intuitive idea to set $\alpha([k, a]) = Tf_k(\alpha_k(a))$ for each $[k, a] \in A$. The problem is that this does not give a well-defined unique function. It is thus important to show that the successor $\alpha(e)$ of some $e \in A$ is independent of the choice of representative of e.

Working towards this solution we once again turn to Chang [4] for inspiration, which will show us that the intuitive idea described above can be achieved in a somewhat roundabout way. Chang [4] describes defining the successor-structure of the colimit of a co-chain of neighbourhood structures by continuously imposing the (conservative) Chang-condition. We will here use the same idea but in a more general setting, although restricted to the conservative weak Chang-condition. Throughout the proof we will work with formulas instead of explicitly using the definition of a weak Chang-morphism. This approach more or less means that we impose the truth of modal formulas on the colimit. To be able to do this, we will from here on assume that our directed system is elementary and we restrict Λ somewhat.

Assumption 6.7. *All f_{kl} are elementary CPL$(\Lambda, \Sigma, \Gamma)$-embeddings.*

Assumption 6.8. *All $\heartsuit \in \Lambda$ are both unary*[22] *and monotone.*

Recall that the goal is to show that there is a coalgebra map $\alpha : A \to TA$ on the carrier set of the colimit so that the maps $f_k : A_k \to A$ are elementary CPL-embeddings. As the proof thereof goes by induction, we basically want the following: if a modal formula is true of some tuple of elements in a certain model in the directed system then the same modal formula is true of the image of that tuple in the colimit $\mathcal{A}$. To do this, we work with a set of *conditions* that a suitable $\alpha : A \to TA$ should meet. To formulate these conditions, consider the following example of a single condition, where we for a moment zoom in on neighbourhood models.

Example 6.9. *Let $(\mathcal{A}_k)_{k\in K}$ be a directed system of monotone neighbourhood models ($T = \mathcal{M}$) and let $(\Lambda, \Sigma, \Gamma) = (\{\Box\}, \{P\}, \emptyset)$, where P is unary. Assume there is some model $\mathcal{A}_k$, for $k \in K$, with some element $a \in A_k$ such that*

$$\mathcal{A}_k \models a\Box\lceil y : y\Box\lceil z : P(z)\rceil\rceil. \tag{34}$$

Naturally we want that

$$\mathcal{A} \models [k, a]\Box\lceil y : y\Box\lceil z : P(z)\rceil\rceil.$$

However, $[|y\Box\lceil z : P(z)\rceil|]^y_{\mathcal{A}}$ depends on $\alpha : A \to TA$, which is by this point not yet defined. Now since we know that all $f_{kk'}$ are elementary CPL-embeddings we do know that

$$\mathcal{A}_{k'} \models f_{kk'}(a)\Box\lceil y : y\Box\lceil z : P(z)\rceil\rceil \text{ for all } k' \geq k. \tag{35}$$

[22] This is just for convenience.

Using the semantics for $\Box$, statement (35) *translates to*

$$\alpha_{k'}(f_{kk'}(a)) \in [|\Box|]_{A_{k'}}\big([|y\Box\ulcorner z : P(z)\urcorner|]^y_{\mathcal{A}_{k'}}\big) \text{ for all } k' \geq k.$$

The induction in the main theorem will prove that ***all*** $f_{k'}$ *are elementary CPL-embeddings so we define the following subset of the domain A of the colimit.*

$$D := \bigcup_{k' \geq k} f_{k'}[|y\Box\ulcorner z : P(z)\urcorner|]^y_{\mathcal{A}_{k'}}$$

We then want that α meets the condition that

$$\alpha([k,a]) \in [|\Box|]_A(D). \tag{36}$$

This is because in the induction we will be able to show that
$(*)$ $D \subseteq [|y\Box\ulcorner z : P(z)\urcorner|]^y_{\mathcal{A}}$. *If we have* (36) *then by monotonicity of $\Box$ applied to* $(*)$ *we get*

$$\alpha([k,a]) \in [|\Box|]_A([|y\Box\ulcorner z : P(z)\urcorner|]^y_{\mathcal{A}}) \text{ so that } \mathcal{A} \models [k,a]\Box y : \big\lceil\Box\ulcorner z : P(z)\urcorner\big\rceil,$$

as desired.

Before we give the formal definitions, let us also illustrate how formulating a condition works in general, going through the steps in the example a bit quicker. We thus turn back to our arbitray directed system $(\mathcal{A}_k)_{k\in K}$ of CPL(Λ,Σ,Γ)-models. Fix $k \in K$, $a \in A_k$, $\heartsuit \in \Lambda$, $\varphi(\vec{x},y) \in \text{CPL}(\Lambda,\Sigma,\Gamma)$ and a tuple $\vec{a} = a_1,..,a_n$ from A_k such that

$$\mathcal{A}_k \models a\heartsuit\ulcorner y : \varphi(\vec{a},y)\urcorner, \text{ i.e. } \alpha_k(a) \in [|\heartsuit|]_{A_k}([|\varphi(\vec{a},y)|]^y_{\mathcal{A}_k}).$$

Then we ultimately want that

$$\alpha([k,a]) \in [|\heartsuit|]_A([|\varphi(f_k(\vec{a}),y)|]^y_{\mathcal{A}}) \text{ so that } \mathcal{A} \models [k,a]\heartsuit\ulcorner y : \varphi(f_k(\vec{a}),y)\urcorner. \tag{37}$$

Note that we explicitly differentiate y from the other variables to stress that it is the comprehension variable for $\heartsuit$. However, (37) cannot be achieved directly since the meaning of $\varphi(f_k(\vec{a}),y)$ in $\mathcal{A}$ may depend on α. To overcome this circularity, we consider the set

$$D := \bigcup_{k' \geq k} f_{k'}\big[[|\varphi(f_{kk'}(\vec{a}),y)|]^y_{\mathcal{A}_{k'}}\big], \tag{38}$$

Note that D is dependent on $k, a, \heartsuit, \varphi(\vec{x},y)$ and $\vec{a}$. The condition is then that

$$\alpha([k,a]) \in [|\heartsuit|]_A(D) \tag{39}$$

Through the induction we will later show that $D \subseteq [|\varphi(f_k(\vec{a}),y)|]^y_{\mathcal{A}}$. Then, similar to the example above, with (39) and monotonicity of $\heartsuit$ we will get (37). Central in the condition is the set in (38); this inspires the following definition.

Definition 6.10. *Given $k \in K$, $a \in A_k$, n-ary $\heartsuit \in \Lambda$, a formula $\varphi = \varphi(x_1, .., x_n, y) \in CPL(\Lambda, \Sigma, \Gamma)$-formula and a tuple $\vec{a} = a_1, .., a_n$ from A_k, such that*

$$\alpha_k(a) \in \heartsuit_{A_k}([|\varphi(a_1, .., a_n, y)|]^y_{\mathcal{A}_k}),$$

we define the set

$$D_{k,a,\heartsuit,\varphi,\vec{a}} := \bigcup_{k' \geq k} f_{k'}\big[[|\varphi(f_{kk'}(a_1), .., f_{kk'}(a_n), y)|]^y_{\mathcal{A}_{k'}}\big].$$

Letting $k, a, \heartsuit, \varphi$ and $\vec{a}$ vary, we also define the following sets:

$$\begin{aligned}
\mathbb{D}_{k,a} &:= \{D_{k,a,\heartsuit,\varphi,\vec{a}} \mid \heartsuit \in \Lambda, \varphi \in CPL(\Lambda, \Sigma, \Gamma), \vec{a} \in A_k\} \\
\mathbb{D}_k &:= \bigcup_{a \in A_k} \mathbb{D}_{k,a} \\
\mathbb{D} &:= \bigcup_{k \in K} \mathbb{D}_k
\end{aligned}$$

Note that the definitions above build on the fact that we are working with a fixed directed system throughout the section. For arbitrary directed systems, one might add an extra index, e.g. $\mathbb{D}_K$, which we will only use once below (in Lemma 6.20). Also note that the index a in $D_{k,a,\heartsuit,\varphi,\vec{a}}$ refers to the condition connected to it: we say that the condition based on the set $D_{k,a,\varphi,\vec{a}}$ is the following statement:

$$\alpha([k, a]) \in [|\heartsuit|]_A(D_{k,a,\varphi,\vec{a}})$$

Of course, one could also formulate more general conditions such as $\alpha([k, a]) \in [|\heartsuit|](D)$ for arbitrary $D \subseteq A$ but we are only interested in the satisfaction of formulas.

Now that we know what kind of properties we want $\alpha : A \to TA$ to have, we can start looking for solutions.

Proposition 6.11. *Fix $k \in K$ and $a \in A_k$. If $\alpha \in TA$ is of the form $Tf_{k'}(\alpha_l(f_{kk'}(a)))$ where $k' \geq k$ then*

$$\alpha \in [|\heartsuit|]_A(D_{k,a,\heartsuit,\varphi,\vec{a}})$$

for all $D_{k,a,\heartsuit,\varphi,\vec{a}} \in \mathbb{D}_{k,a}$.

Proof. Take any $D_{k,a,\heartsuit,\varphi,\vec{a}} \in \mathbb{D}_{k,a}$; recall that we have

$$D_{k,a,\heartsuit,\varphi,\vec{a}} = \bigcup_{k' \geq k} f_{k'}\big[[|\varphi(f_{kk'}(a_1), .., f_{kk'}(a_n), y)|]^y_{\mathcal{A}_{k'}}\big].$$

for some $\heartsuit$, some $\varphi(\vec{x}, y)$ and some $\vec{a} \in A_k$. Now by the definition of $D_{k,a,\heartsuit,\varphi,\vec{a}}$ we have

$$\begin{aligned}
&\alpha_k(a) \in [|\heartsuit|]_{A_k}([|\varphi(\vec{a}, y)|]^y_{\mathcal{A}_k} \\
&\Rightarrow \mathcal{A}_k \models a \heartsuit \ulcorner y : \varphi(\vec{a}, y) \urcorner && \text{(definition)} \\
&\Rightarrow \mathcal{A}_{k'} \models f_{kk'}(a) \heartsuit \ulcorner y : \varphi(f_{kk'}(\vec{a}), y) \urcorner && (f_{kk'} \text{ is an el. CPL-emb.}) \\
&\Rightarrow \alpha_{k'}(f_{kk'}(a)) \in [|\heartsuit|]_{A_{k'}}([|\varphi(f_{kk'}(\vec{a}), y)|]^y_{\mathcal{A}_{k'}}) && \text{(definition)}
\end{aligned}$$

We thus have $(*)$ $\alpha_{k'}(f_{kk'}(a)) \in [|\heartsuit|]_{A_{k'}}([|\varphi(f_{kk'}\vec{a}), y)|]^y_{\mathcal{A}_{k'}})$. Now consider the following two sets:

$$\begin{aligned}
[|\heartsuit|]_{A_{k'}}\big(\mathcal{Q}f_{k'}(D_{k,a,\heartsuit,\varphi,\vec{a}})\big) &= [|\heartsuit|]_{A_{k'}}\big(\{b' \in A_{k'} |\ f_{k'}(b') \in D_{k,a,\heartsuit,\varphi,\vec{a}}\}\big) \text{ and} \\
\mathcal{Q}Tf_{k'}\big(\heartsuit_A(D_{k,a,\heartsuit,\varphi,\vec{a}})\big) &= \{\alpha_{k'} \in TA_{k'} \mid Tf_{k'}(\alpha_{k'}) \in \heartsuit_{[}|A|](D_{k,a,\heartsuit,\varphi,\vec{a}})\}
\end{aligned}$$

By monotonicity and $(*)$ we have $\alpha_{k'}(f_{kk'}(a)) \in [|\heartsuit|]_{A_{k'}}\big(\mathcal{Q}f_{k'}(D_{k,a,\heartsuit,\varphi,\vec{a}})\big)$. Then we can use the naturality of $\heartsuit$ with respect to f_l:

$$[|\heartsuit|]_{A_{k'}}\big(\mathcal{Q}f_{k'}(D_{k,a,\heartsuit,\varphi,\vec{a}})\big) = \{\alpha_{k'} \in TA_{k'} \mid Tf_{k'}(\alpha_{k'}) \in [|\heartsuit|]_A(D_{k,a,\heartsuit,\varphi,\vec{a}})\} \tag{40}$$

We thus have that $Tf_{k'}(\alpha_{k'}(f_{kk'}(a))) \in [|\heartsuit|]_A(D_{k,a,\heartsuit,\varphi,\vec{a}})$. This was to show. □

Now we know the form of 'solutions' $\alpha \in TA$ that meet specific conditions based on the elements of $\mathbb{D}_{k,a}$ for a fixed element $[k, a] \in A$. Next, we need to show that these solutions are independent of the representative of $[k, a]$, i.e. that they are also solutions of $\mathbb{D}_{l,b}$ with $(l, b) \sim (k, a)$. Then, we can assemble a full map $\alpha : A \to TA$. It is very important to stress that the map $\alpha : A \to TA$ 'generated' by the collection $\mathbb{D}$ does in general not have to be unique. We just need there to be at least one map that 'solves all our problems' so that the induction in the main theorem can work.

However, there is a problem. The above proposition suggests that the *problems* in a given model in the system can be solved by successor-states in any following model. But what about the problems in those 'later' models in the system? Those might make use of parameters that have not occurred before, making it impossible to translate them as problems in any foregoing model. Since this is quite important, let us be a bit more specific. Say that we have a directed system indexed by K, where K is just a chain, and that A is the carrier set of its supposed colimit. We want to show that there is a coalgebra map $\alpha : A \to TA$ and we consider the *successor* of some $[k, a] \in A$. Suppose that we set, just like the proposition above suggests $\alpha([k, a]) = Tf_{k'}(\alpha_{k'}(f_{kk'}(a)))$ for some $k' \geq k$. Now we know by the proposition that this choice of $\alpha([k, a])$ 'fixes' all the problems of all models that come *before* $\mathcal{A}_{k'}$, including $\mathcal{A}'_k$. But what about the problems of models *after* $\mathcal{A}_{k'}$? For the theorem that we want to proof, we want to include all elements in models after $\mathcal{A}_{k'}$ as well. The problem is - we stress it again - that these may not have representatives in $\mathcal{A}_{k'}$. So, without any further assumptions, it is not even possible to show that a suitable coalgebra map exists for the colimit of a chain.

To make up for this shortcoming, we look at a kind of *directed compactness* where we want to jump from having a 'solution' in TA for finitely many representatives of an element of the carrier set of the colimit to having a solution for *all* representatives of an equivalence class. We will describe this in the next section.

6.1.1 Directed Compactness

To fix the problem of the parameters described above, we need an assumption of the following form.

Assumption 6.12 (Directed Compactness)**.** *Let $e \in A$. Assume that for any finite number ($n \in \omega$) of representatives $(k_1, a_1), .., (k_n, a_n)$ of e there is an element $\alpha_{fin}(e) \in TA$ such that*

$$\alpha_{fin}(e) \in \heartsuit_A(D_{k_i, a_i, \heartsuit, \varphi, \vec{a}})$$

for $1 \leq i \leq n$ and for all $\heartsuit \in \Lambda$, $\varphi = \varphi(\vec{x}, y) \in CPL(\Lambda, \Sigma, \Gamma)$ and $\vec{a} \in A_{k_i}$. Then there is an element $\alpha(e) \in TA$ such that

$$\alpha(e) \in \heartsuit_A(D_{k, a, \heartsuit, \varphi, \vec{a}})$$

for all $(k, a) \sim e$ and all $\heartsuit \in \Lambda$, $\varphi = \varphi(\vec{x}, y) \in CPL(\Lambda, \Sigma, \Gamma)$ and $\vec{a} \in A_k$.

It should be clear that the antecedent of the implication that forms this assumption can be quite straightforwardly shown to hold using directedness. The question remains if we can account for/ work with the assumption in some form or other by for example one of the following:

1. Characterise assumption 6.12 in terms of T, or even Λ.
2. Prove that assumption 6.12 holds anyway.
3. Each time some form of directed system is needed, prove that assumption 6.12 holds for that specific directed system (for example systems of finite width or directed systems of the form needed to show Robinson's consistency theorem for CPL).

The last option seems the most feasible candidate out of these three but even for chains we cannot seem to find a proof (yet), due to the problem described in the previous section. Maybe, given some $[k, a]$, there is some kind of limit of the set

$$\{Tf_{k'}(\alpha_{k'}(f_{k'}(b)))) \mid k' \geq l\}.$$

There may also be some ways to lighten the assumption a little bit. If we just have it for chains (directed system of width 1) then we might show it for directed systems of arbitary *finite* width.

For a given directed set, you would have to identify the different (interconnected) chains that form the set and then for each equivalence class collect representatives from each chain so that by directedness we can find a model 'above' them all and base the coalgebra successor on this element.

If it is not possible to prove the assumption or show that it holds for systems of finite width, we need to somehow incorporate it in the statement of the Robinson's Consistency theorem as well as the interpolation theorem. So the version of CPL as a whole (not the functor or the set Λ) needs to have the 'Directed Compactness' property. In a negative perspective, the assumption then boils down to the statement 'colimits work for this CPL-instance'. In the rest of the chapter, we will work with the assumption 6.12, which implies that there is a suitable coalgebra map for the colimit of a directed system of CPL-models. Since the assumption itself is quite detailed we will therefore just say that we assume that 'colimits work' for CPL, assuming the following:

Assumption 6.13. *For each $[k,a] \in A$ there is an element $\alpha \in TA$ of the form $Tf_{k'}(\alpha_l(f_{kk'}(a)))$ such that*

$$\alpha \in \heartsuit_A(D_{l,b,\heartsuit,\varphi,\vec{b}})$$

for all $(l,b) \sim (k,a)$ and all $\heartsuit \in \Lambda$, $\varphi = \varphi(\vec{x},y) \in CPL(\Lambda,\Sigma,\Gamma)$ and $\vec{b} \in A_l$.

This will make the rest of the chapter easier to read.

6.1.2 Main Theorem

Now that we know that our colimit has a coalgebra map with the properties we want, we can prove that the f_k are elementary $\mathrm{CPL}(\Lambda,\Sigma,\Gamma)$-embeddings.

For the quantifier case, we need a short lemma.

Lemma 6.14. *For any $k,l \in K$ with $k \leq l$ and any $a \in A_k$ there is some $b \in A_l$ such that $(k,a) \sim (l,b)$.*

Consider $b = f_{kl}(a)$. We have $l \geq k,l$ and $f_{ll}(f_{kl}(a)) = f_{kl}(a)$ so by definiton $(k,a) \sim (l, f_{kl}(a))$.

Theorem 6.15. *Assume that* 6.13 *holds. If all f_{kl} are elementary CPL(Λ,Σ,Γ)-embeddings, then so are all f_k.*

Proof. It is to show that for all $k \in K$ we have

$$\mathcal{A}_k \models \varphi(a_1,..,a_n) \Leftrightarrow \mathcal{A} \models \varphi(f_k(a_1),..,f_k(a_n)) \tag{41}$$

for all $\vec{a} = a_1,..,a_n \in A_k$ and all $\mathrm{CPL}(\Lambda,\Sigma,\Gamma)$-formulas $\varphi = \varphi(x_1,..,x_n)$. The proof goes by induction on the structure of φ. In Corollary 6.4 we established that, no matter the nature of $\alpha : A \to TA$, all f_k are CPL-embeddings. By triviality of the Boolean cases, we here only cover the modality and the quantifier case.

Modality Case Consider $\psi = x\heartsuit\lceil y : \varphi(x_1, .., x_n, y)\rceil$ for some $\heartsuit \in \Lambda$ and some $\varphi = \varphi(x_1, .., x_n, y)$. We prove both directions of (41) for ψ and arbitrary $\mathcal{A}_k$ and arbitrary $a, a_1, .., a_n \in A_k$.

Left to right Assume $\mathcal{A}_k \models a\heartsuit\lceil y : \varphi(a_1, .., a_n, y)\rceil$. Then $\alpha_k(a) \in \heartsuit_{A_k}([|\varphi(\vec{a}, y)|]^y_{A_k})$. Now consider

$$D_{k,a,\heartsuit,\varphi,\vec{a}} = \bigcup_{k' \geq k} f_{k'}\big[[|\varphi(f_{kk'}(a_1), .., f_{kk'}(a_n), y)|]^y_{\mathcal{A}_{k'}}\big].$$

By assumption 6.13 we have a coalgebra map α such that $(*)$ $\alpha([k, a]) \in \heartsuit_A(D_{k,a,\heartsuit,\varphi,\vec{a}})$. Now the inductive hypothesis for $\mathcal{A}_{k'}$, with $k' \geq k$, $\varphi(\vec{x}, y)$ and $f_{kn}(\vec{a})$ yields

$$d \in [|\varphi(f_{kk'}(a_1), .., f_{kk'}(a_n), y)|]^y_{\mathcal{A}_{k'}} \Leftrightarrow f_{k'}(d) \in [|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}}. \tag{42}$$

Notice here that we used the fact that $f_{k'} \circ f_{kk'} = f_k$. We thus have

$$D_{k,a,\heartsuit,\varphi,\vec{a}} \subseteq [|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}} \tag{43}$$

To see this, let $d \in D_{k,a,\heartsuit,\varphi,\vec{a}}$. Then $d \sim (k', a')$ for some $k' \geq k$ and some $a' \in A_{k'}$ such that $a' \in [|\varphi(f_{kk'}(a_1), .., f_{kk'}(a_n), y)|]^y_{\mathcal{A}_{k'}}$. Then by (42) we have $f_{k'}(a') = d \in [|\varphi([k, a_1, .., [k, a_n], y])|]^y_{\mathcal{A}}$. This finishes the proof of (43). Now by $(*)$, monotonicity of $\heartsuit$ and (43) we have

$\alpha([k, a]) \in \heartsuit_A([|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}})$, which gives $\mathcal{A} \models [k, a]\heartsuit\lceil y : \varphi([k, a_1], ..[k, a_n], y)\rceil$.

This concludes the first direction.

Right to left Assume $\mathcal{A} \models [k, a]\heartsuit\lceil y : \varphi(y, [k, a_1], .., [k, a_n])\rceil$. By assumption 6.13 we know that $\alpha([k, a]) = Tf_l(\alpha_l(b))$ for some $(l, b) \sim (k, a)$. So we have

$$Tf_l(\alpha_l(b)) \in \heartsuit_A([|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}}).$$

Now notice that for $\varphi(\vec{x}, y)$, $\mathcal{A}_l$ and $a_1, .., a_n$ the inductive hypothesis gives

$$b \in [|\varphi(f_{kl}(a_1), .., f_{kl}(a_n), y)|]^y_{\mathcal{A}_l} \Leftrightarrow f_l(b) \in [|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}}. \tag{44}$$

Then by the right-to-left part of (44) we have

$$\mathcal{Q}(f_l\big[[|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}}\big]) \subseteq [|\varphi(f_{kl}(a_1), .., f_{kl}(a_n), y)|]^y_{\mathcal{A}_l}. \tag{45}$$

Furthermore, we have

$$\mathcal{Q}Tf_l\big(\heartsuit_A([|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}}))\big) = \{\alpha_l \in TA_l \mid Tf_l(\alpha_l) \in \heartsuit_A([|\varphi([k, a_1], .., [k, a_n], y)|]^y_{\mathcal{A}})\}. \tag{46}$$

It is clear that $\alpha_l(b)$ is part of the set in (46). Now by naturality of $\heartsuit$ with respect to f_l we know that

$$\heartsuit_{A_l}\big(\mathcal{Q}(f_l\big[[|\varphi([k,a_1],..,[k,a_n],y)|]^y_{\mathcal{A}}\big]\big)\big) = \mathcal{Q}Tf_l\big(\heartsuit_A([|\varphi([k,a_1],..,[k,a_n],y)|]^y_{\mathcal{A}})\big)\big). \quad (47)$$

By the observation that $\alpha_l(b)$ is an element of the set on the right-hand side, it must also be an element of the left-hand-side. This fact, together with monotonicity of $\heartsuit$ and (45) gives

$$\alpha_l(b) \in \heartsuit_{A_l}([|\varphi(f_{kl}(a_1),..,f_{kl}(a_n),y)|]^y_{\mathcal{A}_l})$$

So then $\mathcal{A}_l \models b\heartsuit\lceil y : \varphi(f_{kl}(a_1),..,f_{kl}(a_n),y)\rceil$. Recall that $(l,b) \sim (k,a)$. So there is some $m \geq k,l$ such that $f_{km}(a) = f_{lm}(b)$. And by assumption, f_{lm} and f_{km} are elementary CPL(Λ,Σ,Γ)-embeddings. So we have $\mathcal{A}_m \models f_{lm}(b)\heartsuit\lceil y : \varphi(f_{km}(a_1),..,f_{km}(a_n),y)\rceil$, remembering that $f_{km} \circ f_{kl} = f_{lm}$. We then get $\mathcal{A}_k \models a\heartsuit\lceil y : \varphi(a_1,..,a_n,y)\rceil$, as desired.

Quantifier Case This case is the same as for colimits in FOL model theory. Take $\varphi := \exists x\psi(x,\vec{x})$ and for the first direction assume $A_k \models \exists x\psi(x,\vec{a})$ for any M_k, some $\vec{a} \in A_k$. Then there is some $a \in A_k$ such that $A_k \models \psi(a,\vec{a})$. By the inductive hypothesis we have $\mathcal{A} \models \psi\big(f_k(a), f_k[\vec{a}]\big)$ which gives $\mathcal{A} \models \exists x\psi(x, f_k[\vec{a}])$.

For the other direction, assume $(\star)$ $\mathcal{A} \models \exists x\psi(x,[k,a_1],..,[k,a_n])$ for some $a_1,..,a_n \in A_k$. It is to show that $\mathcal{A}_k \models \exists\psi(x,a_1,..,a_n)$. By $(\star)$, there is some $[k',a'] \in A$ such that $A \models \psi([k',a'],[k,a_1],..,[k,a_n])$. By directedness we have k'' such that $k,k' \leq k''$.

Then by lemma 6.14 there are $b,b_1,..,b_n \in A_{k''}$ such that $(k',a') \sim (k'',b)$ and $(k,a_i) \sim (k'',b_i)$ for $i \leq n$. So we have $\mathcal{A} \models \psi([k'',b],[k'',b_1],..,[k'',b_n])$. By the inductive hypothesis for $\mathcal{A}_{k''}$ we then have $\mathcal{A}_{k''} \models \psi(b,b_1,..,b_n)$ so $\mathcal{A}_{k''} \models \exists x\psi(x,b_1,..,b_n)$. Now recall that Lemma 6.14 gave us $b_i = f_{kk''(a_i)}$ and that $f_{kk''}$ is an elementary CPL-embedding. We thus have $\mathcal{A}_k \models \exists x\psi(x,a_1,..,a_n)$, which was to show. □

Lemma 6.16. *Let $(\mathcal{A}_k)_{k\in K}$ be a directed system of CPL(Λ,Σ,Γ)-models and let $\mathcal{A} = (A,\alpha,I)$ be its colimit. If there is a 'competing' colimit $\mathcal{B} = (B,\beta,J)$ then there is an elementary CPL(Λ,Σ,Γ)-embedding $g : A \to B$.*

Proof. Let g be the map as defined in Proposition 6.5. So $g([k,a]) = g_k(a)$. We know by assumption that each g_k is an elementary CPL(Λ,Σ,Γ)-embedding. First, we have

$$\mathcal{A} \models \varphi([k_1,a_1],..,[k_n,a_n]) \Leftrightarrow \text{there is some } k \geq k_1,..,k_n \text{ and } \mathcal{A} \models \varphi([k,f_{k_1k}(a_1)],..,[k,f_{k_nk}(a_n)])$$

We thus get

$$\begin{aligned}\mathcal{A} \models \varphi([k_1, a_1], .., [k_n, a_n]) &\Leftrightarrow \mathcal{A} \models \varphi([k, f_{k_1 k}(a_1)], .., [k, f_{k_n k}(a_n)]) \\ &\Leftrightarrow \mathcal{A}_k \models \varphi(f_{k_1 k}(a_1), .., f_{k_n k}(a_n)) \\ &\Leftrightarrow \mathcal{B} \models \varphi(g_k(f_{k_1 k}(a_1)), .., g_k(f_{k_n k}(a_n))) \\ &\Leftrightarrow \mathcal{B} \models \varphi(g_{k_1}(a_1), .., g_{k_n}(a_{=} n)) \\ &\Leftrightarrow \mathcal{B} \models \varphi(g([k_1, a_1]), .., g([k_n, a_n]))\end{aligned}$$

We conclude that g is an elementary $\mathrm{CPL}(\Lambda, \Sigma, \Gamma)$-embedding. □

6.2 Build-up Lemmas

We are now at a point where we can and will use the Coalgebraic Diagram ElCoDiag defined in chapter 4. This section gives the three lemmas that we will use in the proof for Robinson's consistency. This is also the place to stress again that from here on we rely heavily on compactness.

Assumption 6.17. *The Λ-structure (T, Λ) has compactness.*

This means that any logic $\mathrm{CPL}(\Lambda, \Sigma, \Gamma)$ arising from the Λ-structure (T, Λ) is compact. Recall from chapter 3 that only the bounded structures satisfy compactness.

Lemma 6.18. *Let $(\Lambda_0, \Sigma_0, \Gamma_0) \subseteq (\Lambda_1, \Sigma_1, \Gamma_1)$ and let $\mathcal{S}_0 = (S_0, \sigma_0, I_0)$ and $\mathcal{S}_1 = (S_1, \sigma_1, I_1)$ be a CPL$(\Lambda_0, \Sigma_0, \Gamma_0)$-model and a CPL$(\Lambda_1, \Sigma_1, \Gamma_1)$-model respectively. If $\mathcal{S}_0 \equiv_{(\Sigma_0, \Lambda_0, \Gamma_0)} \mathcal{S}_1$, then there is some CPL$(\Lambda_1, \Sigma_1, \Gamma_1)$-model $\mathcal{S}$ with $f_0 : S_0 \to S$ and $f_1 : S_1 \to S$ a CPL$(\Lambda_0, \Sigma_0, \Gamma_0)$-embedding and a CPL$(\Lambda_1, \Sigma_1, \Gamma_1)$-embedding respectively*[23].

$$\mathcal{S}_0 \xrightarrow{f_0} \mathcal{S} \xleftarrow{f_1} \mathcal{S}_1$$

Proof. We use the method of CoDiagrams to show this. Assume that $\mathcal{S}_0 \equiv_{(\Sigma_0, \Lambda_0, \Gamma_0)} \mathcal{S}_1$. If we can show that

$$T := \mathrm{ElCoDiag}_{(\Lambda_0, \Sigma_0, \Gamma_0)}(\mathcal{S}_0) \cup \mathrm{ElCoDiag}_{(\Lambda_1, \Sigma_1, \Gamma_1)}(\mathcal{S}_1)$$

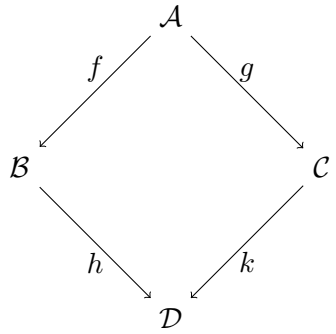
is consistent, then we know that there is a $\mathrm{CPL}(\Lambda_1, \Sigma_1, \Gamma_1)$-model $\mathcal{S}$ as described above. For a reductio, assume that T is inconsistent. By compactness and taking conjunctions, we then get formulas

$$\varphi(\vec{c_a}) \in \mathrm{ElCoDiag}_{(\Lambda_0, \Sigma_0, \Gamma_0)}(\mathcal{S}_0) \text{ and } \psi(\vec{c_b}) \in \mathrm{ElCoDiag}_{(\Lambda_1, \Sigma_1, \Gamma_1)}(\mathcal{S}_1)$$

[23] Note that we leave out the coalgebra structures in the diagram here.

that contradict. We have $\mathcal{S}_1 \models \neg\varphi(\vec{a})$. Now the constants $\vec{c_a}$ do not occur in $\varphi(\vec{x})$, so by Proposition 3.3 we have $\mathcal{S}_1 \models \forall\vec{x}\neg\varphi(\vec{x})$, which implies that $\mathcal{S}_1 \models \neg\exists\vec{x}\varphi(\vec{x})$. By $\mathcal{S}_0 \equiv_{(\Sigma_0,\Lambda_0,\Gamma_0)} \mathcal{S}_1$ then $\mathcal{S}_0 \models \neg\exists\vec{x}\varphi(\vec{x})$, which is a contradiction since $\mathcal{S}_0 \models \varphi(\vec{a})$. We conclude that T is consistent so there is some model $\mathcal{S}$, as desired. □

Lemma 6.19. *Let $(\Lambda_0, \Sigma_0, \Gamma_0) \subseteq (\Lambda_1, \Sigma_1, \Gamma_1)$. Let $\mathcal{A} = (A, \alpha, I_A)$ and $\mathcal{B} = (B, \beta, I_B)$ be two CPL$(\Lambda_0, \Sigma_0, \Gamma_0)$-models and let $\mathcal{C} = (C, \gamma, I_C)$ be a CPL$(\Lambda_1, \Sigma_1, \Gamma_1)$-model. If there are elementary CPL$(\Lambda_0, \Sigma_0, \Gamma_0)$ embeddings $f : A \to B$ and $g : A \to C$, then there is a CPL$(\Lambda_1, \Sigma_1, \Gamma_1)$-model $\mathcal{D} = (D, \delta, I_D)$ with an elementary CPL$(\Lambda_0, \Sigma_0, \Gamma_0)$-embedding $h : B \to D$ and an elementary CPL$(\Lambda_1, \Sigma_1, \Gamma_1)$-embedding $k : C \to D$ such that the diagram below commutes.*



Proof. Both f and g are elementary so $\mathcal{A} \equiv_{(\Lambda_0,\Sigma_0,\Gamma_0)} \mathcal{B}$ and $\mathcal{A} \equiv_{(\Lambda_1,\Sigma_1,\Gamma_1)} \mathcal{C}$. The latter also means $\mathcal{A} \equiv_{(\Lambda_0,\Sigma_0,\Gamma_0)} \mathcal{C}$ and since $\equiv_{(\Lambda_0,\Sigma_0,\Gamma_0)}$ is an equivalence class, we have $\mathcal{B} \equiv_{(\Lambda_0,\Sigma_0,\Gamma_0)} \mathcal{C}$. By Lemma 6.18 we then get the desired $\mathcal{D}$. To make sure that $h \circ f = k \circ g$, we may add constants c_a for each $a \in A$. Take some $a \in A$. Then

$$h(f(a)) = h(f(c_a^{\mathcal{A}})) = h(c_a^{\mathcal{B}}) = c_a^{\mathcal{D}} = k(c_a^{\mathcal{C}}) = k(g(c_a^{\mathcal{A}})) = k(g(a))$$

since all maps are in particular strong CPL-morphisms. □

For the construction in the proof for Robinson's consistency Theorem, we will need the following lemma[24]. Here we will briefly add an extra index to the set $\mathbb{D}$.

Lemma 6.20. *Let $(\mathcal{A}_k)_{k\in K}$ be a directed system of CPL$(\Lambda, \Sigma, \Gamma)$-models. If $J \subseteq K$ is cofinal in K then $(\mathcal{A}_j)_{j\in j}$ and $(\mathcal{A}_k)_{k\in K}$ have isomorphic colimits.*

Proof. We focus on the coalgebra structure here. The important observation here is that $\mathbb{D}_J \subseteq \mathbb{D}_K$. Partially due to assumption 6.12 we know that there is a coalgebra map $\alpha : A \to TA$ that meets all the conditions based on the elements of $\mathbb{D}_K$ so that same map α also meets all the conditions based on the elements of $\mathbb{D}_J$. In short, we know that $(\mathcal{A}_j)_{j\in j}$ and $(\mathcal{A}_k)_{k\in K}$ share a colimit. □

[24]Will need a reference for part of this lemma.

6.3 Robinson's Consistency & Interpolation

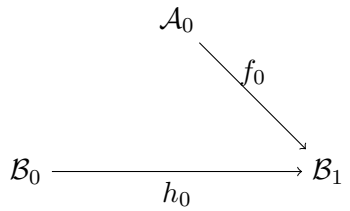
For any two languages $(\Lambda, \Sigma, \Gamma)$ and $(\Lambda', \Sigma', \Gamma')$, let

$$(\Lambda, \Sigma, \Gamma) \cap (\Lambda', \Sigma', \Gamma') := (\Lambda \cap \Lambda', \Sigma \cap \Sigma', \Gamma \cap \Gamma'). \tag{48}$$

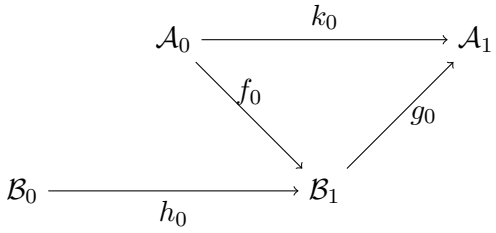
The respective proofs of the following two theorems are more or less the same as the proofs for the same theorems for first-order logic. We want to stress yet again that we work under the assumption that colimits work as described by assumption 6.13.

Theorem 6.21 (Robinson's Consistency Theorem)**.** *Assume that we have 6.13. Let* $(\Lambda_0, \Sigma_0, \Gamma_0)$ *and* $(\Lambda_1, \Sigma_1, \Gamma_1)$ *be two CPL-languages and set* $(\Lambda, \Sigma, \Gamma) := (\Lambda_1, \Sigma_1, \Gamma_1) \cap (\Lambda_0, \Sigma_0, \Gamma_0)$*. Let* T_i *be a* $CPL(\Lambda_i, \Sigma_i, \Gamma_i)$*-theory for* $i = 0, 1$*. Let* T *be a complete* $CPL(\Lambda, \Sigma, \Gamma)$*-theory* T_0 *and* T_1 *both extend. If* T_0 *and* T_1 *are consistent, then so is* $T_0 \cup T_1$*.*

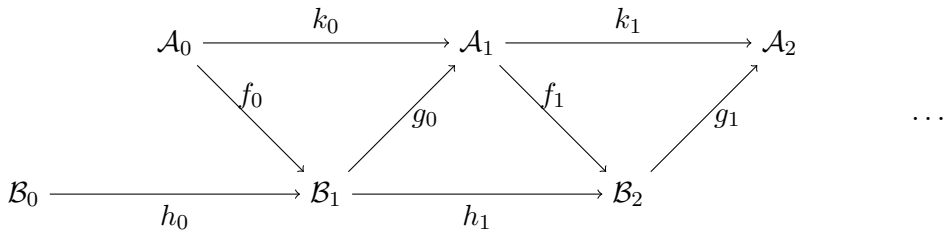
Proof. Assume T_0, T_1 are consistent. Then let $\mathcal{A}_0$ be a model for T_0 and $\mathcal{B}_0$ a model for T_1. It is to show that $T_0 \cup T_1$ has a model. We will do this by building a directed system of CPL-models which as a colimit will have a model that models both T_1 and T_2. We start by noticing that $\mathcal{A}_0, \mathcal{B}_0 \models T$ and since T is complete $\mathcal{A}_0 \equiv_{(\Lambda,\Sigma,\Gamma)} \mathcal{B}_0$. Then by Lemma 6.18 there is a $CPL(\Lambda_1, \Sigma_1, \Gamma_1)$-model $\mathcal{B}_1$ with $f_0 : A_0 \to B_1$ an elementary $CPL(\Lambda, \Sigma, \Gamma)$-embedding and $h_0 : B_0 \to B_1$ an elementary $CPL(\Lambda_1, \Sigma_1, \Gamma_1)$-embedding.



Now by applying Lemma 6.19 to f_0 and the identity on $\mathcal{A}_0$, we get a $CPL(\Lambda_0, \Sigma_0, \Gamma_0)$-model $\mathcal{A}_1$ with $k_0 : A_0 \to A_1$ an elementary CPL$(\Lambda_0, \Sigma_0, \Gamma_0)$-embedding and $g_0 : B_1 \to A_1$ a $CPL(\Lambda, \Sigma, \Gamma)$-embedding.



Notice that with the elementary CPL-embeddings f_0 and k_0 and the fact that $\equiv_{(\Lambda,\Sigma,\Gamma)}$ is an equivalence relation we have $\mathcal{A}_1 \equiv_{(\Lambda,\Sigma,\Gamma)} \mathcal{B}_1$. So we can, just as with $\mathcal{A}_0$ and $\mathcal{B}_0$ in the beginning, apply Lemma 6.18. We continue by again applying Lemma 6.19 and after that iterating the foregoing construction to get a full directed system.



An overview of the models and elementary CPL-embeddings (in what language they are):

- All $\mathcal{A}_i$ are $CPL(\Sigma_0, \Lambda_0, \Gamma_0)$-models.
- All $\mathcal{B}_i$ are $CPL(\Sigma_1, \Lambda_1, \Gamma_1)$-models.
- All f_i and g_i are elementary $CPL(\Sigma, \Lambda, \Gamma)$-embeddings.
- All k_i are elementary $CPL(\Sigma_0, \Lambda_0, \Gamma_0)$-embeddings.
- All h_i are elementary $CPL(\Sigma_1, \Lambda_1, \Gamma_1)$-embeddings.

Now notice that the index sets for the two separate chains consisting of all $\mathcal{A}_i$ and all $\mathcal{B}_i$ are both cofinal in the index set of the whole system. So the colimit $\mathcal{S}$ of the system as a whole is also a colimit of these two chains. We thus have $\mathcal{S} \models T_1 \cup T_2$. □

For a formula $\varphi \in \mathrm{CPL}(\Sigma, \Lambda, \Gamma)$ let CPL_φ be the language consisting of exactly the symbols that occur in φ.

Theorem 6.22 (Craig Interpolation)**.** *Assume that we have 6.13. Let* $(\Lambda, \Sigma, \Gamma)$ *be a language and let* $\varphi, \psi \in CPL(\Lambda, \Sigma, \Gamma)$ *be two sentences. Then there is a sentence* $\theta \in CPL_\varphi \cap CPL_\psi$ *- the interpolant - such that*

$$\varphi \models \theta \text{ and } \theta \models \psi.$$

The proof of this theorem relies heavily on compactness.

Proof. Set $T_\varphi := \{\chi \mid \chi \in \mathrm{CPL}_\varphi \cap \mathrm{CPL}_\psi \text{ and } \varphi \models \chi\}$. We claim that $T_\varphi \models \psi$:

- Suppose not. Then $T_\varphi \cup \{\neg\psi\}$ is consistent and we have a model $\mathcal{S} \models T_\varphi \cup \{\neg\psi\}$. Take $T_\mathcal{S} := \mathrm{Th}_{(\Lambda,\Sigma,\Gamma)}(\mathcal{S})$ (so $T_\varphi \subseteq T_\mathcal{S}$). Claim within claim: $T_\mathcal{S} \cup \{\varphi\}$ is consistent.
  - If not, there is some $\chi \in T_\mathcal{S}$ such that $\varphi \models \neg\chi$. Then $\neg\chi \in T_\varphi \subseteq T_\mathcal{S}$. Contradiction.

  Still within the first claim, still assuming that $T_\varphi \not\models \psi$, we have that $T_\mathcal{S}$ is the theory of a model so $T_\mathcal{S}$ is complete. Furthermore, $T_\mathcal{S} \cup \{\neg\psi\}$ and $T \cup \{\varphi\}$ are consistent. Then by Theorem 6.21 above, we have that $T_\mathcal{S} \cup \{\varphi, \neg\psi\}$ is consistent, contradicting the initial assumption that $\varphi \models \psi$. So we must have $T_\varphi \models \psi$.

By compactness, there are $\theta_1, .., \theta_n \in T_\varphi$ such that $\theta_1, .., \theta_n \models \psi$. Set $\theta := \bigwedge_{i \leq n} \theta_i$. Then θ is the interpolant since we have $\theta \models \psi$ and we obviously have $\varphi \models \theta$. $\square$

We conclude this chapter by stating that the assumption of directed compactness and what it might depend on has to be explored further if one wishes to prove interpolation using the approach that we have offered in this chapter. In the next chapter, we will take a syntactic approach towards interpolation.

7 Proof Theoretic Interpolation

Switching to the proof theory side of things, this chapter does two things. First, it proves interpolation for a specific instance of CPL: neighbourhood frames. Second, and maybe more importantly, it sketches a road map for proving interpolation for arbitary instances of CPL. Along the way, we employ Maehara's method and set up some helpful definitions.

7.1 Set-up and Maehara's Method

The approach towards interpolation is different from the one in the previous chapter so it is good to remind us what interpolation is.

Definition 7.1. *A logic L has interpolation if for any two L-sentences φ and ψ such that $\varphi \models_L \psi$ there is a sentence θ such that $\varphi \models_L \theta$ and $\theta \models_L \psi$ and $L(\theta) \subseteq L(\varphi) \cap L(\psi)$.*

Here $L(\varphi)$ denotes the set of symbols occurring in φ. Note that this definition is already different from the notion of interpolation used in theorem 6.22 in the foregoing chapter. The reason is that we want to be a bit more precise with the definition of $L(\varphi)$ in our syntactic description. And if we have a sound and complete proof system for a logic and we can prove interpolation for that proof system, then we have interpolation for the logic as defined above. In the next section, we will define interpolation with some more detail.

For the more detailed syntactic definition of interpolation, we need the following auxiliary definition.

Definition 7.2. *The set $\Sigma(\varphi)$ of first-order predicates occurring in a CPL$(\Lambda, \Sigma, \Gamma)$-formula φ is defined inductively:*

$$\begin{aligned}
\Sigma(\bot) &= \emptyset \\
\Sigma(s = t) &= \emptyset \\
\Sigma(P(t_1, .., t_n)) &= \{P\} \\
\Sigma(\varphi * \psi) &= \Sigma(\varphi) \cup \Sigma(\psi) \text{ for } * \in \{\vee, \to\} \\
\Sigma(\exists x \varphi) &= \Sigma(\varphi) \\
\Sigma(x \heartsuit \lceil y : \varphi \rceil) &= \Sigma(\varphi)
\end{aligned}$$

Given a set Φ of CPL$(\Lambda, \Sigma, \Gamma)$, we set

$$\Sigma(\Phi) = \bigcup_{\varphi \in \Phi} \Sigma(\varphi).$$

The set $iVar(t)$ of variables occurring in a CPL$(\Lambda, \Sigma, \Gamma)$-term is defined by:

$$\begin{aligned}
iVar(t) &= \{x\} && \text{if } t = x \text{ for some } x \in iVar \\
iVar(t) &= \emptyset && \text{if } t = c \text{ for some } c \in \mathcal{L}
\end{aligned}$$

We then define the set $FV(\varphi)$ of free variables occurring in a CPL$(\Lambda, \Sigma, \Gamma)$-formula φ:

$$\begin{aligned} FV(\bot) &= \emptyset \\ FV(s = t) &= iVar(s) \cup iVar(t) \\ FV(P(t_1, .., t_n)) &= iVar(t_1) \cup \cdots \cup iVar(t_n) \\ FV(\varphi * \psi) &= FV(\varphi) \cup FV(\psi) \text{ for } * \in \{\vee, \to\} \\ FV(\exists x\varphi) &= FV(\varphi) - \{x\} \\ FV(x\heartsuit\lceil y : \varphi\rceil) &= (FV(\varphi - \{y\}) \cup \{x\} \end{aligned}$$

Similarly, we define $\Gamma(\varphi)$ to be the set of constants occurring in φ.

Notice that the occurrence of predicate liftings is not included. This is because those have construction rules, as we define below. Furthermore notice that we always have $\Sigma(\varphi) \subseteq \Sigma$ and $\Gamma(\varphi) \subseteq \Gamma$ if φ is a CPL$(\Lambda, \Gamma, \Sigma)$-formula. CPL-derivation systems $\mathcal{SR}$ are defined in the next section.

7.1.1 The Sequent System

Since we use Γ and Σ for the language of CPL, we use Δ, Φ, Ψ etc. for *multisets* of formulas in *sequents*. A multiset is a set with multiplicity and a sequent $\Phi \Rightarrow \Delta$ is an expression with finite multisets Φ and Δ. Since the CPL-syntax includes the FOL-syntax, we first recap the rules of **G1c**[25], as given in [12, p. 61], for the logical connectives and the first-order quantifiers. Let Φ and Δ be arbitary multisets of formulas and let φ and ψ be arbitary formulas[26].

$$\frac{}{\varphi \Rightarrow \varphi}\,\mathrm{Ax} \qquad \frac{}{\bot \Rightarrow}\,\mathbf{L}\bot$$

$$\frac{\Phi \Rightarrow \Delta, \varphi}{\Phi, \neg\varphi \Rightarrow \Delta}\,\mathbf{L}\neg \qquad \frac{\varphi, \Phi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg\varphi}\,\mathbf{R}\neg$$

$$\frac{\Phi \Rightarrow \Delta, \varphi \qquad \psi, \Phi \Rightarrow \Delta}{\varphi \to \psi, \Phi \Rightarrow \Delta}\,\mathbf{L}\to \qquad \frac{\varphi, \Phi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \to \psi}\,\mathbf{R}\to$$

$$\frac{\varphi[z/x], \Phi \Rightarrow \Delta}{\forall x\varphi, \Phi \Rightarrow \Delta}\,\mathbf{L}\forall \qquad \frac{\Phi \Rightarrow \Delta, \varphi[y/x]}{\Phi \Rightarrow \Delta, \forall x\varphi}\,\mathbf{R}\forall\dagger_y$$

Figure 2: **Rules for the Logical Operators and Constants**

[25] The two-sided Gentzen system for classical logic.

[26] All in a language that includes the connectives and quantifiers, of course.

Note that $\dagger_y$ means that y is fresh for the conclusion. We refer to e.g. φ in $\mathbf{L}\neg$ as the *principal formula*. To this, we also add rules for equality.

$$\frac{}{x = x}\ \mathbf{R}=$$

$$\frac{x = y, \Phi[x/z] \Rightarrow \Delta[x/z]}{x = y, \Phi[y/z] \Rightarrow \Delta[y/z]}\ \mathbf{L}=_1$$

$$\frac{x = y, \Phi[y/z] \Rightarrow \Delta[y/z]}{x = y, \Phi[x/z] \Rightarrow \Delta[x/z]}\ \mathbf{L}=_2$$

Figure 3: **Equality Rules**

And we also include the structural rules, both weakening (**W**) and contraction (**C**), are

$$\frac{\Phi \Rightarrow \Delta, \varphi, \varphi}{\Phi \Rightarrow \Delta, \varphi}\ \mathbf{RC} \qquad \frac{\Phi, \varphi, \varphi \Rightarrow \Delta}{\Phi, \varphi \Rightarrow \Delta}\ \mathbf{LC}$$

$$\frac{\Phi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \varphi}\ \mathbf{RW} \qquad \frac{\Phi \Rightarrow \Delta}{\Phi, \varphi \Rightarrow \Delta}\ \mathbf{LW}$$

Figure 4: **Structural Rules**

And finally the cut-rule:

$$\frac{\Phi_1 \Rightarrow \Delta_1, \varphi \qquad \varphi, \Phi_2 \Rightarrow \Delta_2}{\Phi_1, \Phi_2 \Rightarrow \Delta_1, \Delta_2}\ \textsc{Cut}$$

The cut-rule is optional and it is important to state that it does not preserve the *subformula property*. This means that if a derivation uses an application of the cut-rule, it might be the case that not all formulas occurring in said derivation are subformulas of the formulas in the conclusion. This will be important once we introduce Maehara's method.

The rules discussed above include all the non-modal sequent rules also presented in [5, p. 37]. We say 'include' because we added the rules for negation, which will make the proof in section 7.2 a little more convenient.

The modal rules are given in the following way. We take a one-step rule and present it in sequent style[27], after which we *translate* it to a full-fledged rule.

Definition 7.3. *A sequent rule*

[27]Definition 7.3 is based on definition 6.1 in [5, p. 36].

$$\frac{\Phi_1 \Rightarrow \Delta_1 \dots \Phi_k \Rightarrow \Delta_k}{\heartsuit_1 \vec{p}_1, \dots, \heartsuit_n \vec{p}_n \Rightarrow \heartsuit_{n+1} \vec{p}_{n+1}, \dots, \heartsuit_{n+m} \vec{p}_{n+m}}$$

represents a one-step rule $R = \mathbf{A}/\mathbf{P}$ *in sequent format if* $\mathbf{A}$ *is propositionally equivalent to* $\bigwedge_{i=1}^{k}(\bigwedge \Phi_i \to \bigvee \Delta_i)$ *and* $\mathbf{P}$ *is propositionally equivalent to* $(\bigwedge_{j=1}^{n} \heartsuit_j \vec{p}_j) \to (\bigvee_{j=n+1}^{m} \heartsuit_j \vec{p}_j)$.

Given the sequent representation of a one-step rule R, we define $\mathcal{S}(R)$, the rule for the sequent system for CPL:

$$\frac{\Psi, \Phi_1 \sigma_x^y \Rightarrow \Delta_1 \sigma_x^y, \Theta \ \dots \ \Psi, \Phi_k \sigma_x^y \Rightarrow \Delta_k \sigma_x^y, \Theta}{\Psi, z\heartsuit_1 \lceil x_1 : \varphi_1 \rceil, .., z\heartsuit_n \lceil x_n : \varphi_n \rceil \Rightarrow z\heartsuit_{n+1} \lceil x_{n+1} : \varphi_{n+1} \rceil, .., z\heartsuit_{n+m} \lceil x_{n+m} : \varphi_{n+m} \rceil, \Theta} \, \mathcal{S}(R)\dagger_y$$

Here $\lceil x_i : \varphi_i \rceil = \lceil x_i^1 : \varphi_i^1 \rceil \dots \lceil x_i^{\mathrm{ar}\heartsuit} : \varphi_i^{\mathrm{ar}\heartsuit} \rceil$, for $1 \leq i \leq n$, is an abbreviation. So the predicate liftings in $\mathcal{S}(R)\dagger_y$ above are not all unary, as a quick glimpse might suggest, but all have their own arbitary arity. Furthermore, the substitution σ_x^y sends p_i^j to $\varphi_i^j[y/x_i^j] \in \mathrm{CPL}(\Lambda, \Sigma, \Gamma)$. Recall that $\dagger_y$ means that the variable y is fresh for the conclusion. Given a sound one-step rule R, $\mathcal{S}(R)$ preserves validity on any CPL-model [5, p. 37].

The full system, given a set of one-step rules $\mathcal{R}$, is called $\mathcal{SR}$. Here is the definition of interpolation that we use.

Definition 7.4. *A CPL-derivation system* $\mathcal{SR}$ *has interpolation if for any* $\mathcal{SR}$*-provable sequent* $\Phi \Rightarrow \Psi$ *there is a formula (an interpolant)* θ *such that:*

- $\Phi \Rightarrow \theta$ *is derivable*
- $\theta \Rightarrow \Psi$ *is derivable*
- $\Sigma(\theta) \subseteq \Sigma(\Phi) \cap \Sigma(\Psi)$
- $\Gamma(\theta) \subseteq \Gamma(\Phi) \cap \Gamma(\Psi)$
- $FV(\theta) \subseteq FV(\Phi) \cap FV(\Psi)$

As mentioned above, the syntactic proof of interpolation relies on the fact that the proof system - in this case the sequent system $\mathcal{SR}$ - is complete. Luckily, Litak et al. [5] have found a completeness result by translating back and forth to the Hilbert-style calculus $\mathcal{HR}$. This result importantly encapsulates the fact that $\mathcal{SR}$ works without adding a cut-rule. This implies that $\mathcal{SR}$ satisfies the subformula property, which is key for applying Maehara's method.

7.1.2 Maehara's Method

Maehara's method is a way of inductively proving interpolation, constructing the interpolant of a sequent by going through the sequent rules applied in its derivation. Based on an overview by Ono [13], we quickly recap the basics needed for the method. As mentioned above, it is of key importance that a proof system that we apply the method to is cut-free or has cut elimination. We will make clear why after giving the necessary definitions. The method relies on *splitting* sequents, which will make it easier to prove the main theorem.

Definition 7.5. *Given a multiset Φ of formulas, the pair $\langle \Phi_1; \Phi_2 \rangle$ is a partition of Φ when the multiset union of Φ_1 and Φ_2 is Φ.*

Furthermore, $\langle (\Phi_1 : \Delta_1); (\Phi_2 : \Delta_2) \rangle$ is a partition of the sequent $\Phi \Rightarrow \Delta$ if $\langle \Phi_1; \Phi_2 \rangle$ is a partition of Φ and $\langle \Delta_1; \Delta_2 \rangle$ is a partition of Δ.

We also refer to partitions as *splits*. Within the context of Maehara's method we generalise the property described by definition 7.4 in terms of sequent-partitions.

Definition 7.6. *Let the sequent $\Phi \Rightarrow \Psi$ be provable and let $\langle (\Phi_1 : \Psi_1); (\Phi_2 : \Psi_2) \rangle$ be an arbitrary partition for $\Phi \Rightarrow \Psi$. Then there is a formula - the interpolant - θ such that:*

1. $\Phi_1 \Rightarrow \theta, \Psi_1$ *is derivable*
2. $\theta, \Phi_2 \Rightarrow \Psi_2$ *is derivable*
3. $\Sigma(\theta) \subseteq \Sigma(\Phi_1, \Psi_1) \cap \Sigma(\Phi_2, \Psi_2)$
4. $\Gamma(\theta) \subseteq \Gamma(\Phi_1, \Psi_1) \cap \Gamma(\Phi_2, \Psi_2)$
5. $FV(\theta) \subseteq FV(\Phi_1, \Psi_1) \cap FV(\Phi_2, \Psi_2)$

Definition 7.4 is of course the instance of definition 7.6 where in the split of the sequent we have $\Phi_2, \Psi_1 = \emptyset$. As noted in [13, p. 242], 'the reason why we need to consider arbitrary partitions [...] comes from forms of rules for implication and negation'. More intuitively: the reason that we want to prove this stronger, more general version comes down to *induction loading*. What we prove in the induction is stronger then what you actually need so that the induction hypothesis that we can use in each step is also strong.

As already mentioned above, Maehara's method goes by induction. On the depth of a derivation in the sequent system, to be precise. You start with a partition of the final sequent of a derivation and work your way up the derivation tree. A full proof therefore boils down to looking at all the rules of the system, where for each rule you move the partition in the lower sequent to the upper sequent(s) so that you can use the induction hypothesis. Furthermore, each rule (or case) breaks down into sub-cases, given by all the possibilities of taking a split of the concluding sequent. How these sub-cases work is made clear in the main proof of the next section (7.2). Now moving the partition of a sequent up

in the derivation tree does not work when we allow for the cut-rule. This is because there is no way of deciding where to put the principal formula of the cut-rule in the partition of the upper sequent. For a cut-free system we know that all formulas occurring in the upper sequent are sub-formulas of formulas in the lower sequent so there is a systematic way of putting for example the two conjuncts of a conjunction in the partition of the upper sequent(s) so that one can apply the induction hypothesis. In the main proof in the next section we will each time make clear how a partition of a lower sequent translates to a partition in the upper sequent.

Due to the inductive nature of the method, applying Maehara's method to a sequent system $\mathcal{SR}$, boils down to separate cases for each $R \in \mathcal{R}$. This is because we already know that Maehara's method can be applied to the rest of the rules, proofs for which can be found in [13, sect. 6.1] and [12, p. 118]. Therefore we can in the next section focus on a single proof-rule. We will go a bit more in-depth on the general approach in the last section of this chapter. However, let us first look at our specific interpolation result for the monotone neighbourhood functor.

7.2 The Monotone Neighbourhood Functor

In this section we use Maehara's method to prove interpolation for monotone neighbourhood frames, yielded by the Λ-structure $\mathcal{M}$ (with predicate lifting $\Box$). To complete the CPL-language, take arbitary Σ and Γ. As noted in chapter 3 we have the following one-step rule for $\Box$, capturing monotonicity and axiomatising the logic. Here p and q are some schematic variables.

$$\frac{p \Rightarrow q}{\Box p \Rightarrow \Box q}\ (\text{Mon.})$$

With definition 7.3, while we also add contexts Ψ and Δ, we get the CPL-version:

$$\frac{\Psi, \varphi[y/z] \Rightarrow \psi[y/z'], \Delta}{\Psi, x\Box\lceil z : \varphi\rceil \Rightarrow x\Box\lceil z' : \psi\rceil, \Delta}\ \mathcal{S}(\text{Mon.})\dagger_y$$

We denote the proof-system for this logic given by the Λ-structure $(\mathcal{M}, \{\Box\})$ as $\mathcal{SR}_{\mathcal{M}}$. As we noted above, we know that all the rules of $\mathcal{SR}_{\mathcal{M}}$ except $\mathcal{S}$(Mon.) are already proven in the literature to admit Maehara's method. Furthermore, thanks to results in [5] we know that $\mathcal{SR}_{\mathcal{M}}$ is a sound and complete sequent system. By these facts, the proof of the theorem below, focuses on the monotonicity rule only. We furthermore use the fact that we prove the auxiliary definition 7.6 instead of definition 7.4.

Theorem 7.7. *$\mathcal{SR}_{\mathcal{M}}$ has interpolation.*

Proof. As mentioned above, we focus on the one sequent-style rule $\mathcal{S}$(Mon.). So assume that

$$\Psi, x\Box\lceil z : \varphi\rceil \Rightarrow x\Box\lceil z' : \psi\rceil, \Delta$$

is the final sequent of a cut-free proof, derived by applying the rule $\mathcal{S}$(Mon.) as above:

$$\frac{\Psi, \varphi[y/z] \Rightarrow \psi[y/z'], \Delta}{\Psi, x\Box\lceil z : \varphi\rceil \Rightarrow x\Box\lceil z' : \psi\rceil, \Delta}\,\mathcal{S}(\text{Mon.})\dagger_y$$

This gives some partition of the lower sequent, which also give rise to a partition of $\Psi \Rightarrow \Delta$. So let $\langle(\Psi_1 : \Delta_1); (\Psi_2 : \Delta_2)\rangle$ be an arbitary but fixed split of the sequent $\Psi \Rightarrow \Delta$. We can then go through the four possibilities for taking a partition of $\Psi, x\Box\lceil z : \varphi\rceil \Rightarrow x\Box\lceil z' : \psi\rceil, \Delta$ as a whole.

1. Take $\langle(\Psi_1, x\Box\lceil z : \varphi\rceil : \Delta_1, x\Box\lceil z' : \psi\rceil); (\Psi_2 : \Delta_2)\rangle$. If we bring the partition *without* $x\Box\lceil z : \varphi\rceil$ and $x\Box\lceil z' : \psi\rceil$ to the upper sequent, we get $\langle(\Psi_1, \varphi[y/z] : \Delta_1, \psi[y/z']); (\Psi_2 : \Delta_2)\rangle$. Then by the induction hypothesis there is a formula θ such that

$$(1a) \qquad \Psi_1, \varphi[y/z] \Rightarrow \theta, \psi[y/z'], \Delta_1 \text{ and}$$
$$(1b) \qquad \theta, \Psi_2 \Rightarrow \Delta_2$$

   are provable. We can apply the Monotonicity rule to sequent (1a):

$$\frac{\Psi_1, \varphi[y/z] \Rightarrow \theta, \psi[y/z'], \Delta_1}{\Psi_1, x\Box\lceil z : \varphi\rceil \Rightarrow \theta, x\Box\lceil z' : \psi\rceil, \Delta_1}\,\mathcal{S}(\text{Mon.})\dagger_y$$

   Now together with sequent (1b) we see that θ is also the interpolant for the lower sequent. First, we do have to check that y is fresh in the conclusion in the above application of the rule. Already including the occurrences of constants, the induction hypothesis also gives:

$$(1c) \qquad \Gamma(\theta) \subseteq \Gamma(\Phi_1 \cup \{\varphi[y/z]\}, \Delta_1 \cup \{\psi[y/z']\}) \cap \Gamma(\Phi_2, \Delta_2)$$
$$(1d) \qquad FV(\theta) \subseteq FV(\Phi_1 \cup \{\varphi[y/z]\}, \Delta_1 \cup \{\psi[y/z']\}) \cap FV(\Phi_2, \Delta_2)$$

   Now because y is fresh in the conclusion in the original application of $\mathcal{S}$(Mon.), it does not occur in Δ so it does not occur in Δ_2. Therefore by (1d) we know that y cannot occur freely in θ. So having θ in the conclusion in the application of $\mathcal{S}$(Mon.) to (1a) does not interfere with the fact that y is fresh for the conclusion.

   We now explicitly cover case 3 of definition 7.6, for which we want that

$$\Sigma(\theta) \subseteq \Sigma(\Psi_1, x\Box\lceil z : \varphi\rceil, x\Box\lceil z' : \psi\rceil, \Delta_1) \cap \Sigma(\Psi_2, \Delta_2). \tag{49}$$

   By the induction hypothesis we already know that

$$\Sigma(\theta) \subseteq V(\Psi_1, \varphi[y/z], \psi[y/z'], \Delta_1) \cap \Sigma(\Psi_2, \Delta_2) \tag{50}$$

   and we see that there are no new predicates in the conclusion so we have (49). Lastly, it is easy to notice that we added no constants or variables to θ so the clauses 4 and 5 of defintion 7.6 are covered.

2. Take $\langle(\Psi_1, x\Box\lceil z : \varphi\rceil : \Delta_1); (\Psi_2 : \Delta_2, x\Box\lceil z' : \psi\rceil)\rangle$. By moving the split without $x\Box\lceil z : \varphi\rceil$ and $x\Box\lceil z' : \psi\rceil$ up to the upper sequent, we get the partition $\langle(\Psi_1, \varphi[y/z]\rceil : \Delta_1); (\Psi_2 : \Delta_2, \psi[y/z']\rceil)\rangle$. By the induction hypothesis with this partition we get an interpolant θ and the derivable sequents (2a) and (2b):

$$(2a) \qquad \Psi_1, \varphi[y/z] \Rightarrow \Delta_1, \theta$$
$$(2b) \qquad \Psi_2, \theta \Rightarrow \Delta_2, \psi[y/z']$$

We also know by the induction hypothesis that

$$(2c)\ FV(\theta) \subseteq FV(\Psi_1 \cup \{\varphi[y/z]\}, \Delta_1) \cap FV(\Psi_2, \Delta_2 \cup \{\psi[y/z']\})$$

Unlike in the previous case, we now know that y *can* occur freely in θ. We thus write $\theta(y)$. We then derive the following two sequents by applying the monotonicity rule to both sequents (2a) and (2b). The variable y in θ will then be bound in the conclusion so that it still does not occur freely in the lower sequent: it is still fresh. From (2a), we get

$$\frac{\Psi_1, \varphi[y/z] \Rightarrow \Delta_1, \theta(y)}{\Psi_1, x\Box\lceil z : \varphi\rceil \Rightarrow \Delta_1, x\Box\lceil y : \theta\rceil}\ \mathcal{S}(\text{Mon.})\dagger_y$$

and from (2b) we get

$$\frac{\Psi_2, \theta(y) \Rightarrow \Delta_2, \psi[y/z']}{\Psi_2, x\Box\lceil y : \theta\rceil \Rightarrow \Delta_2, x\Box\lceil z' : \psi\rceil}\ \mathcal{S}(\text{Mon.})\dagger_y$$

We thus get the interpolant $x\Box\lceil y : \theta\rceil$. In the case that y does not occur freely in θ the binding of y in the conclusion is vacuous. Furthermore, we have

$$\Sigma(x\Box\lceil w : \theta\rceil) = \Sigma(\theta) \subseteq \Sigma(\Psi_1, \varphi, \Delta_1) \cap \Sigma(\Psi_2, \psi, \Delta_2).$$

by the inductive hypothesis. So

$$\Sigma(x\Box\lceil w : \theta\rceil) \subseteq \Sigma(\Psi_1, x\Box\lceil z : \varphi\rceil, \Delta_1) \cap \Sigma(\Psi_2, x\Box\lceil z' : \psi\rceil, \Delta_2),$$

as desired. Similar reasoning can be used for the free variables and the constants occurring in the interpolant.

3. Take $\langle(\Psi_1 : \Delta_1, x\Box\lceil z' : \psi\rceil); (\Psi_2, x\Box\lceil z : \varphi\rceil : \Delta_2)\rangle$. Once again, we move up the split to the upper sequent to the partition $\langle(\Psi_1 : \Delta_1, \psi[y/z']); (\Psi_2, \varphi[y/z] : \Delta_2)$. By the induction hypothesis we then get an interpolant θ for this partition, yielding the provable sequents (3a) and (3b), where we again notice that y may occur freely in θ.

$$(3a)\ \ \Psi_1 \Rightarrow \Delta_1, \psi[y/z'], \theta(y)$$
$$(3b)\ \ \Psi_2, \theta(y), \varphi[y/z] \Rightarrow \Delta_2$$

From (3a) we may derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\Psi_1 \Rightarrow \Delta_1, \psi[y/z'], \theta(y)}{\Psi_1, \neg\theta(y) \Rightarrow \Delta_1, \psi[y/z']}\,(L\neg)}{\Psi_1, x\Box\lceil y : \neg\theta\rceil \Rightarrow \Delta_1, x\Box\lceil z' : \psi\rceil}\,\mathcal{S}(\text{Mon.})\dagger_y}{\Psi_1 \Rightarrow \Delta_1, x\Box\lceil z' : \psi\rceil, \neg x\Box\lceil y : \neg\theta\rceil}\,(R\neg)$$

and from (3b) we may derive

$$\dfrac{\dfrac{\dfrac{\dfrac{\Psi_2, \theta(y), \varphi[y/z] \Rightarrow \Delta_2}{\Psi_2, \varphi[y/z] \Rightarrow \Delta_2, \neg\theta(y)}\,(R\neg)}{\Psi_2, x\Box\lceil z : \varphi\rceil \Rightarrow \Delta_2, x\Box\lceil y : \neg\theta\rceil}\,\mathcal{S}(\text{Mon.})\dagger_y}{\Psi_2, x\Box\lceil z : \varphi\rceil, \neg x\Box\lceil y : \neg\theta\rceil \Rightarrow \Delta_2}\,(L\neg)$$

By the two last sequents in the two derivations, we get the interpolant $\neg x\Box\lceil y : \neg\theta\rceil$, which one might see as the *dual* form of the interpolant in case 2. Checking the symbol-occurrences of the interpolant is similar to case 2.

4. Take $\langle(\Psi_1 : \Delta_1); (\Psi_2, x\Box\lceil z : \varphi\rceil : \Delta_2, x\Box\lceil z' : \psi\rceil)\rangle$. This case is very similar to the first one, only the (Mon.)-rule can now be applied to the *second* sequent given by moving the sequent up and the corresponding induction hypothesis, instead of the first.

The case for the monotonicity rule concludes the induction. □

With this, we have proven interpolation for Chang's neighbourhood logic. Of course, it was very important that the monotonicity rule axiomatises modal neighbourhood logic, making the single rule strongly one-step complete for this logic.

7.3 General Interpolation Results: A Blueprint

Now that we have proven interpolation for a specific CPL-sequent system, the question rises whether this result can be generalised. This section briefly sketches - part of the section is intentionally not too specific on details - a road map for general interpolation results via the same method as in the previous section.

The first step towards understanding how Maehara's method can be applied to a general sequent system $\mathcal{SR}$ can be broken down into three points. (1) We will be looking for interpolation results for sequent systems that are sound and complete. (2) A proof for interpolation for a sequent system $\mathcal{SR}$ via Maehara's method has as many cases as there are proof-rules in $\mathcal{R}$. (3) It is more convenient to work with the sequent representation of one-step rules than within the full CPL-environment. Summarising the three points: if one can show that Maehara's method is applicable to a set of (sequent representations of) one-step-rules $\mathcal{R}$, then the instance of CPL that $\mathcal{SR}$ is complete for has interpolation. Let us formulate this a bit clearer.

Definition 7.8. *Let* $\mathcal{S}(R)$

$$\frac{\Phi_1 \Rightarrow \Delta_1 \dots \Phi_k \Rightarrow \Delta_k}{\heartsuit_1 \vec{p}_1, \dots, \heartsuit_n \vec{p}_n \Rightarrow \heartsuit_{n+1} \vec{p}_{n+1}, \dots, \heartsuit_{n+m} \vec{p}_{n+m}} \mathcal{S}(R)$$

be the sequent representation of the one-step rule $R = \mathbf{A}/\mathbf{P}$ *and let* $I \subseteq \{1, .., n\}$ *and* $J \subseteq \{n+1, .., n+m\}$ *so that*

$$\langle (\{\heartsuit_i \vec{p_i}\}_{i \in I} : \{\heartsuit_j \vec{p_j}\}_{j \in J}); (\{\heartsuit_i \vec{p_i}\}_{i \notin I} : \{\heartsuit_j \vec{p_j}\}_{j \notin J}) \rangle$$

is an arbitary partition of the lower sequent in $\mathcal{S}(R)$*.*

We say that $\mathcal{S}(R)$ *is Maehara-friendly if there is a way to move the partition to the upper sequent so that we can derive an interpolant* θ *for the partition of the lower sequent that depends on the formulas in the upper sequent.*

The definition of a Maehara-friendly (MF) proof rule can of course be generalised to any kind of proof rule Of course the idea of Maehara's method is application to full proof systems but this definition hopefully shines a light on what is necessary for the method to work for one-step rules in general.

For the third point above, one could, for even more simplicity in the induction steps of the Maehara-proof, attempt to prove a lemma like the following.

Lemma 7.9. *Let* R *be any sound one-step rule. If* R *is Maehara-friendly then so is* $\mathcal{S}(R)$*.*

This would be technically challenging with a lot of notational detail so we leave it as an open research question for now. We do want to note that one might want to switch to a one-sided sequent system for the CPL proof system, due to the technical complexity of a general one-step rule.

As the three steps given above are basically a formal breakdown of what is necessary in an application of Maehara's method to a complete and cut-free proof system, we now turn to a broader perspective of what might be possible in interpolation results for CPL. We sketch an ideal plan towards showing interpolation for general CPL instances which, in conjunction with section 7.2, this might be helpful for future research.

Figure 7.3 below shows the *blueprint* or road map of steps that one might take in order to show interpolation of an arbitary instance of CPL. Note that one could try to show interpolation for CPL in general but it might very well be possible that some instances of CPL just don't have interpolation as a logic. Given a set of predicate liftings Λ, we denote the coalgebraic modal logic with the elements of Λ in its syntax as ML_Λ and the CPL instance with Λ as CPL_Λ. We then start our road map towards interpolation for CPL_Λ with the already quite strong assumption that ML_Λ has the Craig interpolation property (CIP). The first objective is to find a MF proof system for ML_Λ. With the three steps given in the beginning of this section, we can then get interpolation for CPL_Λ.

1. ML_Λ has the CIP
2. ML_Λ has a 'Maehara-friendly' (MF) proof system
3. CPL_Λ has a MF proof system
4. CPL_Λ has the CIP

Figure 5: **Tentative Road Map**

So we want to show $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4$. The most challenging step would probably be $1 \Rightarrow 2$. However, as we already noticed, 1 is still a pretty strong assumption and it is only natural to also look for restrictions there. In other words: if these 4 steps are provable we might start playing with the first assumption and see what is possible. Again, this is not something the current thesis sets out to do and we conclude the body of the thesis here. We have in this last section merely described a possible direction that a general proof might go. We thus end the thesis with an outlook for further research.

8 Conclusion

This thesis has contributed to the field of coalgebraic predicate logic (CPL) by expanding on what has been done in the literature as well as looking back at the origins of CPL. The most straightforward result is perhaps the development of an Ehrenfeucht-Fraïssé game for CPL and its adequacy theorem. More promising and interesting for further research however, are the explorations in the approaches towards interpolation results for CPL. The thesis focused on interpolation by looking at both sides of CPL.

On the semantic side we analysed whether the construction of colimits of FOL can be adapted to CPL, showing that this is only possible under a certain assumption. This assumption is one of the things that could be further researched, as we briefly described in chapter 6. Of course, there might very well be other approaches towards semantic interpolation that have not yet been found.

On the syntactic side, we have obtained interpolation for the CPL instance given by the monotone neighbourhood functor $\mathcal{M}$. The next step is possibly interpolation for the general neighbourhood functor $\mathcal{N}$, as well as other set functors. For this, we have provided a road map for the syntactic method via Maehara's method, breaking down what is needed for the proof to work as well as offering a list of steps building on interpolation for coalgebraic modal logic.

The results in this thesis thus suggest that there is quite a lot yet to be done in the field of CPL. Almost all chapters offer ideas for more generalisation - think of EF-games that include non-unary predicate liftings, for example, as well as the road map for proof theoretic interpolation, which can be both fleshed out and tried on different instances of CPL. Future research might also look at questions like the importance of monotonicity in for example the EF-game for CPL as we have given it. Anyway, we want to end on a positive note on CPL as a promising field, for which this thesis has given some helpful steps in the right direction.

References

[1] A. Kurz and J. Velebil, "Relation lifting, a survey," *Journal of Logical and Algebraic Methods in Programming*, vol. 85, no. 4, pp. 475–499, 2016, Relational and algebraic methods in computer science, ISSN: 2352-2208. DOI: https://doi.org/10.1016/j.jlamp.2015.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352220815000802.

[2] R. A. Leal, "Modalities through the looking glass: A study on coalgebraic modal logics and their applications," Institute for Logic, Language and Computation., 2011.

[3] C. Kupke and D. Pattinson, "Coalgebraic semantics of modal logics: An overview," *Theoretical Computer Science*, vol. 412, no. 38, pp. 5070–5094, 2011, CMCS Tenth Anniversary Meeting, ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2011.04.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397511003215.

[4] C. C. Chang, "Modal model theory," *Cambridge Summer School in Mathematical Logic*, vol. Volume 337 of LNM, pages 599-617. Springer, 1973.

[5] T. Litak, D. Pattinson, K. Sano, and L. Schröder, "Model Theory and Proof Theory of Coalgebraic Predicate Logic," *Logical Methods in Computer Science*, vol. Volume 14, Issue 1, Mar. 2018. DOI: 10.23638/LMCS-14(1:22)2018. [Online]. Available: https://lmcs.episciences.org/4390.

[6] F. Seifan, L. Schröder, and D. Pattinson, "Uniform Interpolation in Coalgebraic Modal Logic," in *7th Conference on Algebra and Coalgebra in Computer Science (CALCO 2017)*, F. Bonchi and B. König, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 72, Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 21:1–21:16, ISBN: 978-3-95977-033-0. DOI: 10.4230/LIPIcs.CALCO.2017.21. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/8041.

[7] B. van den Berg, "Syllabus model theory 2021/2022," Institute for Logic, Language and Computation, University of Amsterdam, 2022.

[8] L. Schröder, D. Pattinson, and T. Litak, "A Van Benthem/Rosen theorem for coalgebraic predicate logic," *Journal of Logic and Computation*, vol. 27, no. 3, pp. 749–773, Jul. 2015, ISSN: 0955-792X. DOI: 10.1093/logcom/exv043. eprint: https://academic.oup.com/logcom/article-pdf/27/3/749/13694161/exv043.pdf. [Online]. Available: https://doi.org/10.1093/logcom/exv043.

[9] D. Marker, *Model Theory : An Introduction*, ser. Graduate Texts in Mathematics. Springer New York, 2002, ISBN: 9780387987606. [Online]. Available: https://books.google.nl/books?id=QieAHk--GCcC.

[10] J. Väänänen, "Generalized quantifiers," in *Models and Games*, ser. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2011, pp. 283–352. DOI: 10.1017/CBO9780511974885.011.

[11] P. Blackburn, J. van Benthem, and F. Wolter, *Handbook of Modal Logic*, ser. ISSN. Elsevier Science, 2006, ISBN: 9780080466668.

[12] A. Troelstra and H. Schwichtenberg, *Basic Proof Theory*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2000, ISBN: 9780521779111. [Online]. Available: https://books.google.nl/books?id=x9x6F%5C_4mUPgC.

[13] H. Ono, "Proof-theoretic methods in nonclassical logic–an introduction," *Theories of types and proofs*, vol. 2, pp. 207–254, 1998. DOI: 10.2969/msjmemoirs/00201C060.

[14] E. Riehl, *Category Theory in Context*, ser. Aurora: Dover Modern Math Originals. Dover Publications, 2017, ISBN: 9780486820804. [Online]. Available: https://books.google.nl/books?id=6B9MDgAAQBAJ.

[15] Y. Venema, "Coalgebra and modal logic: An introduction," Institute for Logic, Language and Computation, University of Amsterdam, 2019.

A Appendix

This appendix covers some general basics that we expect the reader to be familiar with, focusing on introducing coalgebras and predicate liftings.

Let us first establish some notational choices. We use $|A|$ for the cardinality of a set and $\subseteq_\omega$ for *finite* subsets. Tuples of elements or variables are written as e.g. $\vec{a}$ and, with a slight abuse of notation, we write e.g. $\vec{a} \in A$ when we mean $a_1, .., a_n \in A$.

A.1 First-order Logic Model Theory

As the biggest part of the CPL-syntax comes from FOL, we record some definitions here[28]. This is especially to stress our treatment of variables and assignments. We thus skip syntax and go straight to the semantics. Without going into detail, a model $\mathcal{A} = (A, I)$ for a language $\mathcal{L}$ - where $\mathcal{L}$ consists of predicates P and constants[29] c - has interpretations for all symbols in $\mathcal{L}$. First the semantics of terms:

Definition A.1. *Let $A = (A, I)$ be a first-order model for a language $\mathcal{L}$ and let $v : iVar \to A$ be a variable assignment. We interpret terms as follows:*

$$t^{\mathcal{A},v} = v(x) \qquad \text{if } t = x \text{ for some } x \in iVar$$
$$t^{\mathcal{A},v} = I(c) \qquad \text{if } t = c \text{ for some } c \in \mathcal{L}$$

For the semantics of formulas, we give, to be complete, *all* the logical connectives. Although we do not include all connectives in the syntax of CPL, we sometimes use the other ones as abbreviations.

Definition A.2. *Let $\mathcal{A} = (A, I)$ be a model for a language $\mathcal{L}$ and let $v : iVar \to A$ be an assignment. We have:*

$$\begin{array}{lcl}
\mathcal{A}, v \models \bot & \Leftrightarrow & \text{never} \\
\mathcal{A}, v \models s = t & \Leftrightarrow & s^{\mathcal{A},v} = t^{\mathcal{A},v} \\
\mathcal{A}, v \models P(\vec{t}) & \Leftrightarrow & \vec{t} \in I(P) \\
\mathcal{A}, v \models \varphi \wedge \psi & \Leftrightarrow & \mathcal{A}, v \models \varphi \text{ and } \mathcal{A}, v \models \psi \\
\mathcal{A}, v \models \varphi \vee \psi & \Leftrightarrow & \mathcal{A}, v \models \varphi \text{ or } \mathcal{A} \models \psi \\
\mathcal{A}, v \models \varphi \to \psi & \Leftrightarrow & \text{if } \mathcal{A}, v \models \varphi \text{ then } \mathcal{A}, v \models \psi \\
\mathcal{A}, v \models \exists x \varphi(x) & \Leftrightarrow & \text{there is some } a \in A \text{ such that } \mathcal{A}, v[a/x] \models \varphi(x) \\
\mathcal{A}, v \models \forall x \varphi(x) & \Leftrightarrow & \text{for all } a \in A \text{ we have } \mathcal{A}, v[a/x] \models \varphi(a)
\end{array}$$

[28] We draw from [9] as well as [7].

[29] For simplicity, we work without functions throughout the thesis so there is no reason to include them here.

We write e.g. $v[a/x]$ to mean that v is modified by mapping x to a. And we often leave out the explicit mention of a variable assignment: so we write e.g $t^{\mathcal{A}}$ to mean $t^{\mathcal{A},v}$ or $\mathcal{A} \models \varphi(a)$ when we mean $\mathcal{A}, v[a/x] \models \varphi(x)$.

Very important for a logic are the properties of (soundness and) completeness and compactness. For a logic L, we use $\models_L$ for the semantic relation.

Definition A.3. *Let L be a logic with a proof system $\vdash_L$. Let φ be an arbitrary L-formula.*

$$\begin{array}{rl} \textit{Soundness:} & \vdash_L \varphi \ \Rightarrow \models_L \varphi \\ \textit{Completeness:} & \models_L \varphi \Rightarrow \vdash_L \varphi \end{array}$$

We usually just say that proof system is complete for a logic if we mean that it is sound *and* complete. Here $L(\varphi)$ denotes the symbols occurring in φ. Interpolation can of course be seen both semantically or syntactically.

Definition A.4. *A logic L is compact if every finitely satisfiable set of L-formulas is satisfiable.*

Compactness can often be shown through completeness with respect to a finitary proof-system.

A.2 Category Theory

See e.g. Riehl [14] for an extensive account. Here we just list the definitions needed in the thesis.

Definition A.5. *A category $\mathbb{C}$ consists of two classes:*

- $\mathbb{C}_0$*: A collection of objects; $X, Y, Z, ...$*
- $\mathbb{C}_1$*: A collection of morphisms: $f, g, h, ...$*

such that:

- *Each morphism has a domain and a codomain; e.g. $f : X \to Y$ means that f has domain X and codomain Y*
- *For each object X there is an identity morphism $1_X : X \to X$*
- *If $f : X \to Y$ and $g : Y \to Z$ then there is a composite morphism $g \circ f : X \to Z$*
- *For any $f : X \to Y$, $1_Y \circ f = f \circ 1_X = f$*
- *Composition is associative: i.e. $f \circ (g \circ h) = (f \circ g) \circ h$ for any f, g, h*

Examples of categories are topological spaces with continuous maps as morphisms and groups with group homomorphisms. However, besides one or two exceptions we mostly work within the category SET, where objects are sets and morphisms are functions between sets.

An important notion is, given a category $\mathbb{C}$, that of the *dual* or *opposite* category $\mathbb{C}^{\text{op}}$, where all domains and co-domains of all morphisms are flipped.

Definition A.6. *A functor $F : \mathbb{C} \to \mathbb{D}$ between two categories $\mathbb{C}$ and $\mathbb{D}$ consists of:*

- *F_0: mapping each object $C \in \mathbb{C}_0$ to some $D \in \mathbb{D}_0$*
- *F_1: mapping each morphism $f \in \mathbb{C}_1$ to some $F_1(g) \in \mathbb{D}_1$, respecting (co-)domains, identities and compositions, i.e. $F_1(1_X) = 1_{F_0(X)}$ and $F_1(f \circ g) = F_1 f \circ F_1 g$. This is called functoriality.*

When the context is clear, we will usually leave out indices for functors and categories, writing e.g. FC for some object $C \in \mathbb{C}$. It is important to distinguish between *covariant* and *contravariant* functors. On morphisms, a contravariant functor $F : \mathbb{C} \to \mathbb{D}$ does the following differently from definition A.6:

- mapping each morphism $f : X \to Y$ in $\mathbb{C}$ to $F(f) : FY \to FX$ such that $F(g \circ f) = F(f) \circ F(g)$ for all $f, g \in \mathbb{C}_1$

Contravariant functors reverse the direction of composition; they can also be seen as covariant on the opposite category.

Another special class of functors are endo-functors: functors whose domain coincides with its codomain. Since we work with SET mostly, here are some examples of endo-functors an the category SET, which we also refer to as *set-functors*.

- The constant functor Δ_C for some set C

On objects:	For all sets A, $\Delta_C(A) = C$
On morphisms:	For all functions f, $\Delta_C(f) = 1_C$

- The *covariant* power-set functor $\mathcal{P}$

On objects:	For all sets A, $\mathcal{P}A := \{X \mid X \subseteq A\}$ is the power-set of A
On morphisms:	For all $f : A \to A'$ and for all $C \subseteq A$, $\mathcal{P}f(C) = \{f(c) \mid c \in C\}$

- The *contravariant* power-set functor $\mathcal{Q}$

On objects:	For all sets A, $\mathcal{Q}A = \mathcal{P}A$
On morphisms:	For all $f : A \to A'$ and for all $C' \subseteq A'$, $\mathcal{Q}f(C') = \{a \in A \mid f(a) \in C\}$

- The bag-functor $\mathcal{B}$

On objects: $\mathcal{B}A = \{\mu : A \to \omega + 1 \mid \mu \text{ is a function }\}$, the set of weight functions on A
On morphisms: Given $f : A \to B$ and $\mu \in \mathcal{B}C$: $(\mathcal{B}f)(\mu)(b) := \Sigma\{\mu(a) \mid a \in A, f(b) = a\}$

For $\mathcal{P}f(C)$ and $\mathcal{Q}f(C')$ we also write $f[C]$ and $f^{-1}[C']$, respectively. We have one last example, building more functors with already defined ones:

- Multiplication of functors $T = F_1 \times \cdots \times F_n$

On objects: For all sets A, $TA = F_1A \times \cdots \times F_nA$
On morphisms: Given $f : A \to B$ and $c \in F_iA$ for $1 \leq i \leq n$, we have $Tf(c) = F_if(c)$

Definition A.7. *Let $F, G : \mathbb{C} \to \mathbb{D}$ be two functors between the categories $\mathbb{C}$ and $\mathbb{D}$. A natural transformation $\mu : F \Rightarrow G$ is a collection of maps, $(\mu_C)_{C \in \mathbb{C}_0}$ with $\mu_C : FC \to GC$, such that the diagram*

$$\begin{array}{ccc} FC & \xrightarrow{\mu_C} & GC \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FC' & \xrightarrow[\mu_D]{} & GC' \end{array}$$

commutes for every $f : C \to C'$ in $\mathbb{C}$.

The commutation of a diagram as in definition A.7 is also referred to as *naturality*.

A.3 Coalgebra

Coalgebras are the dual of algebras, having a *composite successor* for each element.

Definition A.8. *Let T be a set-functor. Given a set A, a coalgebra is a pair $(A, \alpha : A \to TA)$.*

The coalgebra map $\alpha : A \to TA$ of a coalgebra (A, α) may be seen as a transition function, associating each $a \in A$ with a composite state, or simply a successor, $\alpha(a) \in TA$. If T is understood, we usually just write (A, α). Furthermore, we refer to the set A as the carrier set or the domain. And we often use corresponding Greek letters for the coalgebra map on a set, i.e. $(A, \alpha), (B, \beta), (C, \gamma)$.. are used to denote coalgebras.

If $T = F_1 \times \cdots \times F_n$ where F_i is a set functor ,for $1 \leq i \leq n$, we use indices to indicate which part of the coalgebra map $\alpha : A \to TA$ we mean, e.g. $\alpha(a)(i)$ denotes the i-th component, given by F_i, of the successor of a.

Definition A.9. *A coalgebra morphism between coalgebras (A, α) and (B, β) is a map $f : A \to B$ such that the following square commutes.*

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ TA & \xrightarrow[Tf]{} & TB \end{array}$$

Or in an equation, we have $\beta \circ f = Tf \circ \alpha$.

Given the definitions A.8 and A.9, we have the category Coalg of coalgebras and coalgebra morphisms[30].

Next, we have the interpretations of modal operators in CPL: predicate liftings.

Definition A.10. *Let T be a set-functor and let $n \in \omega$. An n-ary predicate lifting is a natural transformation $\lambda : \mathcal{Q}^n \Rightarrow \mathcal{Q} \circ T^{op}$.*

Let T be a set-functor and let λ be an n-ary predicate lifting. For any coalgebra[31] (A, α) we thus have a map

$$\lambda_A : \mathcal{Q}^n A \to \mathcal{Q}TA$$

Modal operators $\heartsuit$ in CPL are interpreted by predicate liftings, denoted by $[|\heartsuit|]$. We look at whether

$$\alpha(a) \in [|\heartsuit|]_A(C) \tag{51}$$

for any $a \in A$, $C \subseteq A$.

Definition A.11. *An n-ary predicate lifting λ is monotone in its ith component if for all sets A we have*

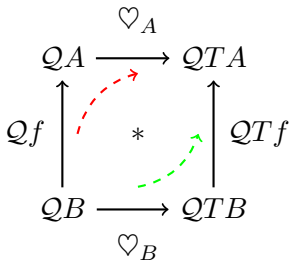
$$\lambda_A(C_1, .., C_i, .., C_n) \subseteq [|\heartsuit|]_A(C_1, .., C_i', .., C_n)$$

for all $C_1, .., C_i, C_i', .., C_n \subseteq A$ such that $C_i \subseteq C_i'$.

For unary predicate liftings, we say monotone (instead of monotone in the first component). Now since we use the naturality of predicate liftings quite a bit in the thesis, let us break it down a little. Let $\lambda : \mathcal{Q} \Rightarrow \mathcal{Q} \circ T^{\mathrm{op}}$ be a unary predicate lifting, let (A, α) and (B, β) be two T-coalgebras and let $f : A \to B$ be a coalgebra morphism. Then the following diagram commutes.

[30] It is quite straightforward to check that this satisfies definition A.5.

[31] Of course for any *set* but we are interested in coalgebra here.



Now take some $D \in QB$ and set $C = f^{-1}[D]$. When we chase the diagram in the two directions, we get

$$\begin{aligned} \text{Red Path}: &\quad D \mapsto \{a \in A \mid f(a) \in D\} = C \mapsto \lambda_A(C) \\ \text{Green Path}: &\quad D \mapsto \lambda_B(D) \mapsto \{\alpha \in TA \mid Tf(\alpha) \in \lambda_B(D)\} \end{aligned}$$

We then have

$$\lambda_A(C) = \{\alpha \in TA \mid Tf(\alpha) \in \lambda_B(D)\}. \tag{52}$$

A.4 One-step Logic

One-step logic is very important for the proof systems of CPL. As put aptly in [15, sect. 7-2], it boils down to 'doing coalgebraic logic without coalgebras'. For one-step logic, one only needs a single element in TA (not a full coalgebra map) and modal operators are interpreted as predicate liftings (as in 51 above)[32].

First, fix a set Λ of modal operators. Given a set of propositional variables V, we define $\mathrm{Prop}(V)$, the set of *rank*-0-*formulas* over V:

$$\pi ::= a \mid \bot \mid \top \mid \neg\pi \mid \pi_0 \vee \pi_1$$

where $a \in V$. Next, given a set $\mathcal{C}$, define

$$\Lambda(\mathcal{C}) := \{\heartsuit(\varphi_1, .., \varphi_n) \mid \heartsuit \in \Lambda \text{ is } n\text{-ary}, C_1, .., C_n \in C\}.$$

It is often assumed that $\mathcal{C}$ is a set of rank-0-formulas over a set of propositional variables V. So then

$$\mathrm{Rank1}(V) = \mathrm{Prop}(\Lambda(\mathrm{Prop}(V)))$$

is the set of set of *rank*-1-*formulas* over V, where each variable falls under the scope of exactly one modal operator $\heartsuit \in \Lambda$.

Now fix a set of *schematic* variables $sVar$. A *one-step rule* is of the form $R = \mathbf{A}/\mathbf{P}$, where $A \in \mathrm{Prop}(sVar)$ and $P \in \mathrm{Rank1}(sVar)$ is a disjunctive clause. Following [5] it is furthermore very important that every $p \in sVar$ may only occur in $\mathbf{P}$ once and that if p

[32] We very closely follow two overviews of one-step logic: the one given in [5] and the one in [15].

occurs in $\mathbf{A}$, it must also occur in $\mathbf{P}$. A one-step rule R is sound when, if the premise $\mathbf{A}$ is true in a model[33] then so is the conclusion $\mathbf{P}$. Now we can also define the last definition of this appendix.

Definition A.12. *A set of predicate liftings Λ is separating if for all coalgebras (A, α) every $\alpha \in TA$ is uniquely determined by the set*

$$\{\heartsuit C_1, .., C_n \in \Lambda(\mathcal{P}S) \mid \alpha \in [|\heartsuit|]_S(C_1, .., C_n)\}.$$

Note that we can equivalently say that given two distinct $\alpha_0, \alpha_1 \in TA$, one can always find some $\heartsuit \in \Lambda$ and some subsets $C_1, .., C_n$ of A such that $\alpha_0 \in [|\heartsuit|]_A(C_1, .., C_n)$ and $\alpha_1 \notin [|\heartsuit|]_A(C_1, .., C_n)$.

[33]One-step frames are sets with a single coalgebra successor. A one-step model additionally has an assignment of the propositional variables V.