Expressivity of Coalgebraic Modal Languages

MSc Thesis (*Afstudeerscriptie*)

written by

Raul Andres Leal Rodriguez
(born January 12th, 1982 in Bogotá, Colombia)

under the supervision of **Dr Yde Venema**, and submitted to the Board of Examiners 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:**
August 29, 2007	Dr Yde Venema
	Prof.dr Johan van Benthem
	Dr Clemens Kupke
	Prof.dr Peter van Emde Boas

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

Contents

Chapter 1

Introduction

Through history logic has been used for several purposes, for example as foundation of mathematics or in philosophy to provide an controlled environment of argumentation. In this thesis we are interested in using logic to describe mathematical structures. For example, it is well know that algebraic logic can be used to describe algebraic structures. Birkhoff Theorem clarifies under what conditions a class of algebraic structures can be characterized using algebraic logic. The dual concept of algebraic structure is that of coalgebraic structure. In the last decade there has been a development of different logical languages to describe coalgebraic structures. There is no agreement in which of these language is the most appropriate to describe coalgebras. This disagreement is partially based on the fact that there is not much work comparing these languages. In this thesis we will do some steps on this direction comparing two languages to describe coalgebraic structures.

Coalgebras and algebras are formally dual concepts, i.e. given a functor T, a coalgebra is a function $\alpha : A \longrightarrow TA$ and an algebra is a function $\alpha : TA \longrightarrow A$. Peter Gumm in [6] claims that coalgebras as direct duals of algebras have been in scene for more than 30 years, but did not receive much attention primarily due to the lack of vital examples. The vital examples came from computer science. Various kind of transitions systems, automata and functional programming languages are naturally represented as coalgebraic structures. These new examples demonstrated that a better intuition to understand universal coalgebra is to conceive it as a general and uniform theory to describe dynamic systems.

Our starting point is that of basic modal logic and Kripke structures, frames and models. It is well known that modal logic is an expressive language to talk about Kripke structures or relational structures, see [2]. Using modal logic we can provide an internal local perspective on relational structures, see [2]. Now Kripke frames and Kripke models can be naturally represented in coalgebraic terms. For example, a Kripke frame (A, R) is represented by a function $R : A \longrightarrow \mathcal{P}A$, where $\mathcal{P}$ is the covariant power set functor and $R(s)$ is the the successors of s. Here, we can see that modal logic is a language to talk about coalgebraic structures. This immediately rises interesting questions. Are there other formal languages like basic modal logic for other coalgebras? Is the modal language an isolated language or is it an example of a more general concept? Since there are many languages to talk about coalgebras what is the relation between them?

The first question can be answered through abstract model theory. A language for coalgebras is a set with a class of satisfaction relations. The satisfaction of a modal formula on some point of a Kripke frame can be seen as a process executed over some Kripke structure. Here, we can say that from the point of modal logic Kripke structures are dynamic systems, therefore coalgebras are a generalization of Kripke frames, but there is no natural interpretation of basic modal formulas over arbitrary coalgebras. Pattinson in [9] and Schröder in [10] solved this problem showing that there is a direct generalization of modal logic, see Chapter 4 here, using the concept of relation lifting; we call those languages *coalgebraic modal languages*. These language can be uniformly defined for a large class of functors. Hence modal logic is a particular instance of a more general phenomenon. Historically, coalgebraic modal languages were not the first languages invented to talk about coalgebras in a uniform framework and as a generalization of modal logic. The first language with this features was invented by Lawerence S. Moss, see [8]; we call this language *Moss' language*. The presence of at least two different languages to talk about coalgebras explain the third question. What is the relation between them? We have no general answer for this question. Out here we do answer the third question in the particular case of Moss' language and coalgebraic modal languages.

To return to our starting point, it is usually claimed that in the particular case of the covariant power set functor, Moss' language and the coalgebraic modal language are equally expressive, see [12]. Unfortunately in the literature used for this thesis, there is no much material explaining what it means for two languages to be equally expressive. This thesis will not provide such general theory. Instead we will compare Moss' language and coalgebraic modal languages at a semantical level and at a syntactical level.

One way to define the expressiveness of a language is using its semantics. A language $\mathcal{L}_1$ is said to be more expressive than a language $\mathcal{L}_2$ if $\mathcal{L}_1$ can express differences between coalgebras that the language $\mathcal{L}_2$ cannot. Using this criterion, we will then show that Moss' language and coalgebraic modal languages are equally expressive.

Another way to compare the expressiveness of two languages is trough translations. Using the notation from the previous paragraph, this means that each formula in the language $\mathcal{L}_2$ is translated into an equivalent formula in $\mathcal{L}_1$. Using this criterion, we will show that in the case of Kripke polynomial functors every predicate lifting can be translated into Moss' language.

The structure of the thesis is as follows: In the proceeding chapter we introduce the formal context in which this thesis is located, we discuss some basics of category theory and universal coalgebra. We also establish the background related to languages and translations, including expressive languages. In Chapter 3 we define Moss' language, provide examples in the case of Kripke polynomial functors, and finally we show how to extend Moss' language with disjunctions and negations. In Chapter 4 we show how to generalize modal logic to coalgebraic modal languages, provide examples in the case of Kripke polynomial functors, and show, in an original work, that coalgebraic modal languages can be represented as initial algebras. With these preliminaries out of the way, we are ready to compare Moss' language and coalgebraic modal languages. In Chapter 5 we demonstrate that the existence of expressive languages is equivalent to the existence of a final object. We present a new elementary proof developed by the author and Clemens Kupke. Using this result, we define non constructive translations between Moss' Language and coalgebraic modal languages. In the final chapter, we refine such translations, defying constructive finitary translations for the particular case of Kripke polynomial functors.

Some final remarks: Unfortunately we have neither space nor time to introduce universal coalgebra from scratch, we assume the reader is familiar with some notions of category theory and coalgebra. The next chapter is only a source of references for this thesis. An introduction like [11] for category theory supplies more than enough material. Another useful source is [4], in this book Robert Golblatt studies the basic concepts in great detail for the particular case of sets. In the area of universal coalgebra there are several different introductions we specially recommend [6], [7] and [12]. As stated, our starting point is that of basic modal logic and Kripke structures. Again there is not enough space to provide a decent introduction to the subject, therefore we assume the reader is familiar with modal logic. We base most of our general knowledge on modal logic in [2]. Finally, we clarify that during this thesis, references in the text are only used to help the reader to find more detailed proofs and explanations. A claim like "Theorem 5.2.1 can be found in [2]" only means that we have used reference [2] as a base for theorem 5.2.1; we do not claim that the original proof or idea first appeared in [2].

Chapter 2

Formalizing the Beginning

In the previous chapter we gave a general introduction to the subject concerning this thesis. This chapter has two objectives. First, to introduce the formal context in that this thesis is placed. As we said before, this thesis is not self contained. This thesis is not an introduction to the theory of coalgebras neither a state of the art in the use of logics for coalgebras. We give no introduction to modal logic, the work done in [2] provides more than enough material. The first section is a short reference to category theory and the second section is a short introduction to coalgebra. The well informed reader can skip these sections without problem. The second objective of this chapter is to make explicit the basic framework that we are gonna use to compare Moss' language and coalgebraic modal languages.

2.1 A Short Reference for Category Theory

This thesis is a friendly interaction between category theory and set theory. In most situations we try to present the subjects from both perspectives but there are plenty of situations where one overwhelms the other and then it makes no sense to consider both perspectives.

We assume familiarity with set theoretical operations, with the concepts of cardinal and ordinal numbers and with the notion of *regular cardinals.*

Definition 2.1.1. *A cardinal number κ is said to be regular if its cofinality is κ itself.*

An intuition behind regular cardinal is that they can not be covered using less than κ steps. We will use regular cardinals to bound disjunctions and conjunctions. An example of a regular cardinal is $\aleph_0$. An example of a non regular cardinal is $\aleph_\omega$.

Here we don't assume familiarity with general category theory but familiarity with a categorical view of sets. We assume the reader is familiar with the concepts of products, pullbacks, equalizers, terminal objects and their duals on the category of sets. The work done by Robert Goldblatt in [4] is a nice source to obtain that knowledge. We also assume the reader is aware of the principle of duality, and the adjunction between exponentials and products, i.e there is a natural correspondence

$$\frac{X \times Y \longrightarrow Z}{X \longrightarrow Z^Y}.$$

This thesis is only about set coalgebras, i.e coalgebras for a functor $T : Set \longrightarrow Set$. Therefore we have to assume familiarity with endofunctors over the category Set. We fix some notation.

Notation 2.1.2. *Given a functor T, we write, indiscriminately, T_f or $T(f)$ to indicate the action of T over a function f.*

Definition 2.1.3. *A category $\mathcal{C}$ is locally small iff given any objects X, Y in $\mathcal{C}$ the arrows between X and Y form a set.*

It is well known than the category of sets is locally small.

Among all the endofunctors over the category Set, the following functors will play and important role: (1) The homomorphism functors.

Definition 2.1.4 ($Hom(-,-)$ functors)**.** *Let $\mathcal{C}$ be a locally small category. The functor*

$$Hom_{\mathcal{C}} : \mathcal{C}^{op} \times \mathcal{C} \longrightarrow Set$$

is defined in the following way. On objects: $Hom_{\mathcal{C}}$ maps a pair of objects, X, Y in $\mathcal{C}$, to the set of arrows, $Hom(X, Y)$, between them. On arrows: $Hom_{\mathcal{C}}$ maps a pair of arrows, $f : X' \longrightarrow X; g : Y \longrightarrow Y'$, to the function

$$Hom_C((f, g)) : Hom(X, Y) \longrightarrow Hom(X', Y').$$

This function maps an arrow $h : X \longrightarrow Y$ to the following composite

$$\begin{tikzcd}
X \arrow[r, "h"] & Y \arrow[d, "g"] \\
X' \arrow[u, "f"] \arrow[r, dashed] & Y'
\end{tikzcd}$$

If it is clear what category is involved we will drop the subindex. If the category $\mathcal{C}$ is a category where the arrows are natural transformations we write $Nat(-,-)$ instead of $Hom(-,-)$.

Notice the following two facts: First, if we fix the first component, say with C, in the previous functor we obtain a covariant functor

$$Hom(C, -) : \mathcal{C} \longrightarrow \mathcal{C}.$$

Second, if we fix the second component, say with C, we obtain a contravariant functor

$$Hom(-, C) : \mathcal{C}^{op} \longrightarrow \mathcal{C}.$$

The category Set has a distinguished object 2, a set with two elements. This set is particularly important because the contravariant functor

$$Hom(-, 2) : Set^{op} \longrightarrow Set$$

is isomorphic to the contravariant power set functor. The contravariant power set functor, written $2^{(-)}$ or sometimes $\check{\mathcal{P}}$, maps a set X to its power set $\check{\mathcal{P}}X$. The action on arrows is as follows: An arrow $f : X \longrightarrow Y$ is mapped to its inverse image $f^{-1} : \check{\mathcal{P}}Y \longrightarrow \check{\mathcal{P}}X$. If f is an inclusion map $i : X \longrightarrow Y$, it is mapped to $(-) \cap X : \check{\mathcal{P}}Y \longrightarrow \check{\mathcal{P}}X$. Notice that we have three different notations for the contravariant power set functor,

$$Hom(-, 2) \text{ and } \check{\mathcal{P}}(-) \text{ and } 2^{(-)},$$

we will use the three of theme indiscriminately. The reason for that is that each of them give a different perspective that is useful for different situations.

(2) Another important functor is the covariant power set functor written $\mathcal{P}$. This functor maps an object X to its power set $\mathcal{P}X$. An arrow $f : X \longrightarrow Y$ is mapped to its direct image $\overline{f} : \mathcal{P}X \longrightarrow \mathcal{P}Y$. We recall that the direct image is defined as follows. Given a subset $A \subseteq X$

$$\mathcal{P}(f) = f[A] = \{f(x) \,|\, x \in A\}.$$

With respect to the covariant power set functor it is interesting to remark

Remark 2.1.5. *The covariant power set functor is not isomorphic to any single functor of the form $Hom(D, -)$, but it is isomorphic to a colimit of such functors.*

(3) Functors can be combined using the limits and colimits on the base category, it is done pointwise. We are particularly interested in products and coproducts. Given two functors $T_1, T_2 : Set \longrightarrow Set$ we define the product functor $T_1 \times T_2 : Set \longrightarrow Set$ as follows. An object X is mapped to $T_1X \times T_2X$. An arrow $f : X \longrightarrow Y$ is mapped to the product arrow $(T_1(f), T_2(f))$. The coproduct of two functors is defined following the same idea.

Remark 2.1.6. *The reader with knowledge in category theory should know that products and coproducts of functors are particular cases of limits and colimits in the category Set^{Set}. This category is the category of endo functors on the category of sets with natural transformations as arrows and usual composition. The existence of limits and colimits in Set^{Set} is a particualr instance of the following general result:*

For any locally small category $\mathbb{C}$ the functor category $Set^{\mathbb{C}}$ has all finite limits and colimits.

We refer the reader to **??** *for a more detailed discussion.*

(4) Other functors that will play an important role in this thesis are Kripke polynomial functors. The collection of *Kripke polynomial functors*, KPFs for short, is inductively defined as follows:

$$K := \mathcal{I} \,|\, D \,|\, K_0 + K_1 \,|\, K_0 \times K_1 \,|\, Hom(D, K) \,|\, \mathcal{P}K.$$

Here $\mathcal{I}$ is the identity functor, D is a constant functor with value D, $\mathcal{P}$ is the covariant power set functor, $Hom(D, -)$ is the homomorphism functor, products and coproducts are defined as we explained on the previous paragraph. Replacing $\mathcal{P}$ with the finite power set functor $\mathcal{P}_\omega$, and demanding D to be finite, we obtain the collection of *finitary Kripke polynomial functors.*

Once we assume familiarity with functors we don't have to do much to encourage *natural transformations* to jump to the stage.

Definition 2.1.7. *Let $F, G : \mathbb{C} \longrightarrow \mathbb{D}$ be functors. A natural transformation λ from F to G (denoted $\lambda : F \longrightarrow G$) is a function that assigns to each object A in $\mathbb{C}$ a morphism $\lambda_A : FA \longrightarrow GA$ in $\mathbb{D}$ in such a way that the following naturality condition holds: for each morphism $f : A \longrightarrow B$, the square on the right*

$$\begin{tikzcd}
\mathbb{C} & & \mathbb{D} & \\
A \arrow[d, "f"'] & FA \arrow[rr, "\lambda_A"] \arrow[d, "F_f"'] & & GA \arrow[d, "G_f"] \\
B & FB \arrow[rr, "\lambda_B"'] & & GB
\end{tikzcd}$$

commutes in $\mathbb{D}$.

The idea of the previous schema is that we have something in $\mathbb{C}$ and there are two ways to transfer it into $\mathbb{D}$. A natural transformation is a transformation between such transformations. A component of a natural transformation are called *operators.*

Another fact that we will use later, specially in the last chapter, is that using functors we can transfer and retract natural transformations.

Definition 2.1.8. *If $F, G : \mathbb{C} \longrightarrow \mathbb{D}$ are functors and $\lambda : F \longrightarrow G$ is a natural transformation, then*

- *for each functor $H : \mathbb{D} \longrightarrow \mathbb{E}$, the natural transformation $H(\lambda) : HF \longrightarrow HG$ is defined by*
$$H(\lambda)_A = H(\lambda_A),$$
this natural transformation is the H-transfer of λ.

- *For each functor $H : \mathbb{E} \longrightarrow \mathbb{C}$, the natural transformation $\lambda_H : FH \longrightarrow GH$ is defined by*
$$(\lambda_H)_A = \lambda_{HA},$$
this natural transformation is the H-retract of λ.

In Chapter 3 we will need the concept of equivalence. We will only mention the definition and some results. The interested reader can find more information in [11].

Definition 2.1.9 (Equivalences)**.** *A functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ is* essentially surjective on objects*, if for any object D in $\mathcal{D}$ there exists an object C in $\mathcal{C}$ such that $F(C)$ is isomorphic to D.*

A functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ is said to be an equivalence if it is full, faithful and essentially surjective on objects. Two categories are equivalent if there exists an equivalence between them.

An important characterization of equivalences is given by the following result, a suggestion for a proof can be found in [11].

Theorem 2.1.10. *A functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ is an equivalence iff there exists a functor $G : \mathcal{D} \longrightarrow \mathcal{C}$ and natural transformations*

$$\mu : Id_C \longrightarrow GF \quad \upsilon : Id_D \longrightarrow FG$$

whose components are all isomorphism.

The intuition behind equivalences is that, in general, asking two categories to be isomorphic is a really strong condition. It can be relaxed using equivalences. Two equivalent categories can not be distinguished using the language of category theory. This happens because objects are defined up to isomorphisms.

2.2 A Short Reference for Coalgebras

Coalgebras and Algebras are formally dual concepts. In this thesis we will only study the particular case of coalgebras associated with a functor $T : Set \longrightarrow Set$, i.e set-coalgebras. Assuming this we define coalgebras and algebras:

Definition 2.2.1. *Let a functor $T : Set \longrightarrow Set$ be given.*

- *A coalgebra for T, or a T-coalgebra, is a function $\alpha : A \longrightarrow TA$.*
- *An algebra for T, or a T-algebra, is a function $\alpha : TA \longrightarrow A$.*

The map α is called a structural map.

Notice that more important than the carrier set is the structural map. In most of the cases there is not a unique structural map associated with a set A. Because of that in most situations we will only write the structural map to represent coalgebras. General introductions to the theory of universal coalgebra can be found in [6] or in [7] or in [9].

Many structures on mathematics and computer science can naturally be represented as coalgebras. Our starting point is that of Kripke frames. A Kripke frame (A, R) is represented coalgebraically as follows: A Kripke frame is a coalgebra for the covariant power set functor. We define $\alpha_R : A \longrightarrow \mathcal{P}A$ to be

$$s' \in \alpha(s) \text{ iff } sRs'.$$

Coalgebras for a functor T are not isolated objects there are morphism between them, more formally they from a category. A coalgebraic morphism is defined as follows

Definition 2.2.2. *A morphism $f : \alpha \longrightarrow \beta$ between coalgebras α and β, for a functor T, is a function $f : A \longrightarrow B$ such that the following diagram*

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "\alpha"'] & B \arrow[d, "\beta"] \\
TA \arrow[r, "T(f)"'] & TB
\end{tikzcd}$$

commutes. We call f a coalgebraic morphism. *Dually we define* algebraic morphisms.

The commutativity of the previous means that the states $s, f(s)$ behave in the same way. It is easy to see that in the case of $\mathcal{P}$-coalgebras, Kripke frames, coalgebraic morphism are precisely bounded morphism between Kripke frames. The concept of bisimulation is naturally generalized to arbitrary coalgebras as follows.

Definition 2.2.3. *If (A, α) and (B, β) are coalgebras, then a relation $R \subseteq A \times B$ is a T-*bisimulation *from α to β if there exists a structural map $\rho : R \longrightarrow TR$ such that the following diagram*

$$\begin{tikzcd}
A \arrow[d, "\alpha"'] & R \arrow[l, "\pi_A"'] \arrow[r, "\pi_B"] \arrow[d, "\rho"] & B \arrow[d, "\beta"] \\
TA & TR \arrow[l, "T(\pi_A)"] \arrow[r, "T(\pi_B)"'] & TB
\end{tikzcd}$$

commutes, i.e the projections are T-morphisms. Two states s, s' in coalgebras α and β, respectively, are said to be bisimilar, written $s \underline{\leftrightarrow} s'$, if there exists a bisimulation R between α and β such that $(s, s') \in R$.

Some properties of bisimulations, see [6] for the complete proofs, that will be used later are:

Proposition 2.2.4. *The union of any collection of bisimulations from α to β is a bisimulation from α to β.*

Based on the previous proposition we conclude that for each pair of coalgebras $(A, \alpha), (B, \beta)$ there exists a largest bisimulation from α to β, the union of all the bisimulations from α to β. We call this relation *bisimilarity*. This will be denoted by $\underline{\leftrightarrow}_{\alpha\beta}$ and may be written without subscript if there is no risk of confusion. We also write $(\alpha, s) \underline{\leftrightarrow} (\beta, s')$ when $s \underline{\leftrightarrow}_{\alpha\beta} s'$, and view bisimilarity as a relation between pointed coalgebras. A functor is said to have *transitive bisimilarity* if

$$s \underline{\leftrightarrow} s' \underline{\leftrightarrow} s'' \text{ implies } s \underline{\leftrightarrow} s'.$$

Corollary 2.2.5. *Given pointed coalgebras $(\alpha, s), (\beta, s')$, then $s \underline{\leftrightarrow} s'$ iff there exists a bisimulation, say R, from α to β such that $(s, s') \in R$.*

There is a deep relation between bisimulation and coalgebraic morphisms. Namely, coalgebraic morphisms are functional bisimulations. Formally, it means.

Proposition 2.2.6. *A function $f : A \longrightarrow B$ is a morphism from α to β iff its graph,*

$$G_f = \{(s, f(s)) \mid s \in A\},$$

is a bisimulation.

Unfortunately bisimulations do not behave nicely in all cases, see [9]. There is another concept that allows us to study the behavior of states, namely *behavioral equivalence.*

Definition 2.2.7. *Two states s, s' in coalgebras α, β, respectively, are* behavioral equivalent*, written $s \sim s'$, if there exists another coalgebra γ and coalgebraic morphisms $f : \alpha \longrightarrow \gamma$ and $g : \beta \longrightarrow \gamma$ such that $f(s) = g(s')$.*

Coalgebraic morphisms preserve the behavior of states, then two states are behavioral equivalent if we can find a new state that behave as the both of the initial states. In the general theory of coalgebras this notion is more natural than the notion of bisimulation.

Coalgebras and algebras are not isolated structures, together with coalgebraic and algebraic morphisms form categories.

Notation 2.2.8. *Let a functor $T : Set \longrightarrow Set$ be given.*

- *We write $Coalg(T)$ for the category of T-coalgebras with coalgebraic morphism and usual functional composition.*
- *We write $Alg(T)$ for the category of T-algebras with algebraic morphism and usual functional composition.*

A final coalgebra is a terminal object in a category of coalgebras. It is easy to see that if a final coalgebra exists then two states are behavioral equivalent iff they behave the same over the final coalgebra.

A category of coalgebras has all the colimits that exists in the base category, that because colimits are computed pointwise, see [6]. For example the coproduct of two coalgebras α, β is computed as follows: At a first instance we have the coproduct of the two structural maps

$$[\alpha, \beta] : A + B \longrightarrow TA + TB.$$

The universal property of the coproduct gives us another arrow

$$[T(i_A), T(i_B)] : TA + TB \longrightarrow T(A + B).$$

It is straight forward to check that

$$[T(i_A)\alpha, T(i_B)\beta] : A + B \longrightarrow T(A + B)$$

is the coproduct of α and β in the category $Coalg(T)$. On the other direction it can be shown, see [6], that the carrier of a colimit in $Coalg(T)$ is the same colimit in the base category.

Remark 2.2.9. *In general limits do not exists in categories of coalgebras, even if all of them exists on the base category. The existence of limits is bounded to the preservation of limits of the functor T, see [6]. In this thesis we dont have to deal with limits of coalgebras.*

We finish this section stating an useful result for coalgebras.

Lemma 2.2.10 (Lambek's lemma)**.** *If $\zeta : Z \longrightarrow TZ$ is a final coalgebra, then ζ is an isomorphism.*

2.3 Languages for Coalgebras

As we said before, our starting point is the fact that the basic modal language is a formal language to express facts about $\mathcal{P}$-coalgebras. In this thesis we will show, see Chapter 4, how to generalize the basic modal language to arbitrary set-functors. Generally speaking languages for coalgebras, see [5] or [10], are defined as follows:

Definition 2.3.1 (Languages for Coalgebras)**.** *Let T be a functor and let $\mathcal{L}$ be a set (language). The set $\mathcal{L}$ is a* local language for T-coalgebras *iff there is an operation, $\models$, such that to each T-coalgebra, (A, α), assigns a satisfaction relation $\models_\alpha \subseteq A \times \mathcal{L}$.*

When a pair (s, φ) belongs to $\models_\alpha$ we write $\alpha, s \models \varphi$, or $s \models_\alpha \varphi$, and say: The formula φ is true*, or satisfied, at state s in A.*

This is equivalent to the existence of a function

$$\models : PCoalg(T) \longrightarrow \mathcal{PL},$$

where the domain is the class of all pointed T-coalgebras.

This definition is quite general. Furthermore notice that we do not assume the existence of any logical connectives like disjunctions or modalities, Goldblatt in [5] suggests that better name could be prelogics. Clearly the basic modal language is a local language for $\mathcal{P}$-coalgebras. But there have been several other examples of local languages in the literature for particular cases of functors since long time ago. For example a modal language with polyadic modalities is a local language for the products of the power set functor with itself.

In 1999 Lawrence S. Moss published a paper, see [8], where he explicitly presented a local language that could be uniformly defined for all set functors, actually standard weak pullback preserving functors, this language is presented here in Chapter 3. Another important example of local languages for coalgebras are the languages with coalgebraic modalities, such languages are presented in Chapter 4. The main, and only, subject of this thesis is to compare these two languages at a semantical level and at a syntactical level.

2.3.1 Semantical Comparisons

Our first approach to compare the expressiveness of local languages is done using semantics. The expressiveness of a language is not given for the facts it can express about coalgebras but for those differences that the language can not express. Notice that for every local language we can restrict the function $\models$ to a single pointed coalgebra (α, s). This produces what we call truth classes, formally truth classes are defined as follows.

Definition 2.3.2 (Truth Classes). *Given a local language $\mathcal{L}$, we associate with each pointed coalgebra (α, s) its* Truth Class

$$\mathcal{L}(\alpha, s) = \{\varphi \in \mathcal{L} \mid s \Vdash_\alpha \varphi\}.$$

A set $\Phi \subseteq \mathcal{L}$ is called a truth class if there exists a coalgebra α and a state $s \in A$ such that $\Phi = \mathcal{L}(\alpha, s)$. We write $\mathcal{L}_\models$ for the set of all truth classes.

Using the truth classes of local language $\mathcal{L}$ for T-coalgebras we can define an equivalence relation $\leftrightsquigarrow_\mathcal{L}$, logical equivalence or logically indistinguishability, over the class of all pointed T-coalgebras as follows:

$$(\alpha, s) \leftrightsquigarrow_\mathcal{L} (\beta, s') \text{ iff } \mathcal{L}(\alpha, s) = \mathcal{L}(\beta, s')$$

Using this equivalence relation we can define expressive power as follows

Definition 2.3.3 (Expressive Power). *Let $\mathcal{L}$ be a local language for T-coalgebras. The* expressive power *of $\mathcal{L}$ is the set of equivalence classes of the relation (α, s) and (β, s') are logically indistinguishable for $\mathcal{L}$.*

As we said this thesis does not contain a general theory of expressiveness hence we do not claim the previous definition is the most appropriate, but it is a start. This definition of expressive power is what we would call a semantical definition. Semantical because the expressive power is a set of pointed coalgebras. Using this definition we can give a first method to compare languages.

Definition 2.3.4 (Semantic Comparison of Expressiveness). *Let $\mathcal{L}_1, \mathcal{L}_2$ be local languages. We say that the language $\mathcal{L}_2$ is* more expressive *than the language $\mathcal{L}_1$, iff*

$$\alpha \leftrightsquigarrow_2 \beta \text{ implies } \alpha \leftrightsquigarrow_1 \beta,$$

where $\leftrightsquigarrow_1$ and $\leftrightsquigarrow_2$ are the respective logically equivalence relations. When a language $\mathcal{L}_1$ is more expressive that a language $\mathcal{L}_2$ we write $\mathcal{L}_1 \leq \mathcal{L}_2$. We say that the language $\mathcal{L}_2$ is strictly more expressive *than a language $\mathcal{L}_1$, iff*

$$\mathcal{L}_1 \leq_M \mathcal{L}_2 \text{ and } \mathcal{L}_2 \not\leq_M \mathcal{L}_1$$

This framework to compare expressiveness is formalizing the following intuition: A language $\mathcal{L}_2$ is more expressive than a language $\mathcal{L}_1$ if and only if $\mathcal{L}_2$ can express properties that $\mathcal{L}_1$ can not. For example if ML is the basic modal language and MLE is the basic modal language together with an universal modality. The language MLE is strictly more expressive that ML.

Since MLE is an extension of ML it is clear that it will be more expressive than ML. To see that it is strictly more expressive let $M_1 = \langle\{x, y\}, \emptyset, \emptyset\rangle$ and $M_2 = \langle\{x, y\}, \emptyset, V(p) = \{y\}\rangle$. Clearly $(M_1, x) \leftrightsquigarrow_{ML} (M_2, x)$, but $(M_1, x) \leftrightsquigarrow_{MLE}$ (M_2, x) does not hold.

2.3.2 Adequate and Hennessy-Milner Languages

In this thesis we are only interested in local languages that behave nicely with respect to bisimulations and behavioral equivalence. Those are called expressive languages. The general idea is the following: Assume we have a local language $\mathcal{L}$ and an equivalence relation R over some class of pointed coalgebras. We would like characterize the relation R using $\mathcal{L}$ but also the other way around, i.e characterize the language $\mathcal{L}$ using R. When this is possible we say that $\mathcal{L}$ is expressive for R.

Adequate Languages

Definition 2.3.5 (Adequacy). *Given an equivalence relation, say R, over $PCoalg(T)$. We say that a local language $\mathcal{L}$ is adequate or appropriate* with respect to R *iff equivalence according to R implies equivalence according to $\mathcal{L}$. That is, for any pair of pointed coalgebras, say (α, s) and and (β, s')*

$$(\alpha, s)R(\beta, s') \textit{ implies } (\alpha, s) \leftrightsquigarrow_{\mathcal{L}} (\beta, s')$$

Adequacy says that the language $\mathcal{L}$ can be characterized using R. Conversely it says, if s and s' are not logically equivalent then they can not be equivalent using R. Notice that adequacy is preserved downwards, i.e under less expressive languages. That is if R is an equivalence over the class of pointed coalgebras, and it is the case that $\mathcal{L}_2$ is adequate with respect to R and $\mathcal{L}_1 \leq \mathcal{L}_2$ then $\mathcal{L}_1$ is adequate with respect to R.

Example 2.3.6. *Some basic examples from modal logic are:*

- *The basic modal language is adequate with respect to bisimilarity of Kripke models.*
- *First order logic is not adequate with respect to bisimulation of Kripke models*

Hennessy-Milner Language

The other side of the coin is to give a logical characterization of an equivalence relation, say R, over the class of pointed coalgebras. In order to do that we need the notion of a Hennessy-Milner language.

Definition 2.3.7 (Hennessy-Milner language). *Let R be an equivalence relation over the class of pointed coalgebras. We say that a language $\mathcal{L}$* has the Hennessy-Milner property with respect to R *iff logical equivalence according to $\mathcal{L}$ implies equivalence according to R. That is for any pointed coalgebras, say (α, s) and (β, s'),*

$$(\alpha, s) \leftrightsquigarrow_{\mathcal{L}} (\beta, s') \textit{ implies } (\alpha, s)R(\beta, s').$$

Conversely it says, if s and s' are not R-equivalent then there is a formula that witness it. The Hennessy-Milner property is preserved upwards, i.e over more expressive languages. That is if R is an equivalence over the class of pointed coalgebras and it is the case that $\mathcal{L}_1$ has the Hennessy-Milner property with respect to R and $\mathcal{L}_1 \leq \mathcal{L}_2$ then $\mathcal{L}_2$ has the Hennessy-Milner property with respect to R. Combining adequate and Hennessy-Milner we obtain expressive languages.

Definition 2.3.8 (Expressive language). *Let R be an equivalence class over the class of pointed coalgebras. We say that a language $\mathcal{L}$* is expressive with respect to *R iff it is adequate and has the Hennessy-Milner property with respect to R*

A relation between the expressiveness of a Hennessy-Milner languages and adequate language is:

Theorem 2.3.9. *Fix an equivalence relation R over the class of pointed coalgebras. Then Hennessy-Milner languages for R are semantically more expressive than adequate languages for R, i.e if $\mathcal{L}_1$ is an adequate language and $\mathcal{L}_2$ has the Hennessy-Milner property then $\mathcal{L}_1 \leq \mathcal{L}_2$.*

Now as consequence of the previous theorem we obtain that expressive language can not be differentiated at a semantical level,

Corollary 2.3.10. *Expressive languages can not be differentiated semantically.*

Perhaps expressive languages can be differentiated using some more sophisticated notion. This leads us to translations, or syntactic comparisons of expressiveness.

2.3.3 Translations

In Chapter 3 using the literature we show that Moss language is expressive. In Chapter 4 using the references we show that we can always find a set of predicate liftings such that the language of predicate liftings is expressive. Hence based on the previous corollary we conclude that these two languages are equally expressive at the semantical level. Because of that we will now approach the comparison issue at a syntactical level. The intuition is the following: A language $\mathcal{L}_2$ is more expressive than a language at a syntactical level than a language $\mathcal{L}_1$ iff for every formula $\varphi \in \mathcal{L}_1$ there exists a formula $\psi \in \mathcal{L}_2$ that is equivalent to φ. Notice that this intuition is defining a function from $\mathcal{L}_1$ to $\mathcal{L}_2$.

Definition 2.3.11 (Translations). *Let $\mathcal{L}_1, \mathcal{L}_2$ be local languages. A* translation *from $\mathcal{L}_1$ to $\mathcal{L}_2$ is a function*

$$Trs : \mathcal{L}_1 \longrightarrow \mathcal{L}_2,$$

such that for every pointed coalgebra (α, s) the following holds: For all $\varphi \in \mathcal{L}_1$

$$\alpha, s \models_{\mathcal{L}_1} \varphi \textit{ iff } \alpha, s \models_{\mathcal{L}_2} Trs(\varphi)$$

Clearly the standard translation $ST : ML \longrightarrow FOLR$ is a translation. The characterizing property of translations can also be expressed using satisfaction sets.

Proposition 2.3.12. *Let $\mathcal{L}_1, \mathcal{L}_2$ be local languages for coalgebras. A function $Trs : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ is a translation iff it commutes with validity. That is for every coalgebra α and every formula $\varphi \in \mathcal{L}_1$ the equality*

$$[\varphi]_{1,\alpha} = [Trs(\varphi)]_{2,\alpha}$$

holds, where $[-]_{i,\alpha}; i = 1, 2$ are the appropriate satisfaction sets.

Translations are related to the semantic comparison of expressive power in the following sense.

Proposition 2.3.13. *Let $\mathcal{L}_1, \mathcal{L}_2$ be local languages. If there is a translation $Trs : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ then $\mathcal{L}_1 \leq \mathcal{L}_2$.*

Proof. Assume $(\alpha, s) \leftrightsquigarrow_{\mathcal{L}_2} (\beta, s')$, we want to prove $(\alpha, s) \leftrightsquigarrow_{\mathcal{L}_1} (\beta, s')$. Using the former assumption and the translation we have:

$$\alpha, s \models_{\mathcal{L}_1} \varphi \text{ iff } \alpha, s \models_{\mathcal{L}_2} Trs(\varphi) \text{ iff } \beta, s' \models_{\mathcal{L}_2} Trs(\varphi) \text{ iff } \beta, s' \models_{\mathcal{L}_1} \varphi.$$

this concludes the proof □

Based on the previous result we see that there is, at least, another form to compare languages.

Definition 2.3.14 (Syntactic Comparison of Expressiveness)**.** *Let $\mathcal{L}_1, \mathcal{L}_2$ be local languages. We say that the language $\mathcal{L}_2$ is* more expressive *than the language $\mathcal{L}_1$* via translations*, iff there exists a translation $Trs : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$.*

The previous proposition implies that the syntactic comparison of expressiveness is finer than the semantic comparison of expressiveness, i.e translations may, in principle, see differences that semantics can not.

Based on the results of the previous section in later chapters we will prove that Moss' language and coalgebraic modal languages are indistinguishable at a semantical level, then the main problem to study in this thesis becomes the following:

> Can we define translations between Moss' language and Coalgebraic modal languages?

In other words the main problem of this thesis can be restated as follows:

> Can we express Moss' modality using predicate liftings and boolean operators? Given a predicate lifting, say λ, can we express λ using Moss' modality and boolean operators?

In the last chapter we answer this question in the case of Kripke polynomial functors.

Chapter 3

Moss' Language

In this chapter we will study the language that Lawrence S. Moss published in the paper *Coalgebraic Logic*, see [8], in 1999. As stated in that paper Moss developed a language to express facts about coalgebraic structures that could be defined uniformly for a large class of functors.

In this chapter, we first present some preliminaries, concentrating on two subjects. One, relation liftings. The satisfaction relation associated with Moss' language is given by relation liftings. Two, κ-accessible functors. We will see that this is a condition needed to guarantee that Moss' language has a set formulas and not a class of formulas. With those preliminaries out of the way, we present present a formulations of Moss' language following the work done by Pattinson in [9]. Afterwards, we will compute Moss' language for Kripke polynomial functors to provide some examples and to prepare the field for Chapter 6. We finish this chapter by showing one way to extend Moss' language with negations and disjunctions. An interesting result is that the existence of a translation is represented by the existence of a full and faithful functor and the equivalence of two categories.

3.1 Preliminaries

First we provide some definitions and results that will be used later.

3.1.1 Accessible Functors

As we have said in several occasions, our starting point is that of the covariant power set functor, and the fact that the basic modal language is a local languages for $\mathcal{P}$-coalgebras. It is well known that modal logic does not have the Hennessy-Milner property with respect to the class of all $\mathcal{P}$-coalgebras. Using Lambek's lemma, see 2.2.10 in the previous chapter, it can be shown that the category $Coalg(\mathcal{P})$ has no terminal object. However, the covariant power set functor has a finitary version $\mathcal{P}_\omega$ such that: One, modal logic can be seen as a logic for $\mathcal{P}_\omega$-coalgebras. Two, modal logic has the Hennessy-Milner property with respect to $\mathcal{P}_\omega$-coalgebras. Furthermore the category $Coalg(\mathcal{P}_\omega)$ has a terminal object.

The functor $\mathcal{P}_\omega$ acts on arrows in the same way the functor $\mathcal{P}$ does. This functor maps a set X the finite subsets of X. In this section we will present a generalization of the functor $\mathcal{P}_\omega$ to arbitrary functors and to arbitrary regular cardinals, see definition 2.1.1 on the previous chapter.

The functor $\mathcal{P}_\omega$ is what we call and accessible functor. Hence a generalization of it has to deal with accessibility. Because of that we first define accessible functors.

Definition 3.1.1 (Accessible functors)**.** *Suppose κ is a regular cardinal. A functor $T : Set \longrightarrow Set$ is κ-*accessible*, if the following holds:*

For all sets X and all $x \in TX$ there is a subset $Y_x \subseteq X$ with $|Y_x| < \kappa$ such that $x \in T_i(Y)$, where $i : Y \longrightarrow X$ is the inclusion and the set $T_i(Y)$ is the direct image of Y under the function T_i. A functor is called accessible *iff it is κ-accessible for some regular cardinal κ.*

Unfortunately it is not relevant for this thesis to explain in full detail the ideas used to the generalize the functor $\mathcal{P}_\omega$. We only present what we claim is a generalization:

Definition 3.1.2 (Small T_κ functors)**.** *Let $T : Set \longrightarrow Set$ be a functor and let κ be a regular cardinal. Given a set X, we define*

$$T_\kappa(X) = \bigcup\{T_i(TA) \mid i : A \longrightarrow X, |A| < \kappa\}$$

where i is the inclusion function. Given a function $f : X \longrightarrow Y$ we define $T_\kappa(f) = T_f$.

The small T_κ functors are important because we can prove the following result.

Proposition 3.1.3. *Let $T : Set \longrightarrow Set$ be a functor. Then for each regular cardinal κ, the functor T_κ is a κ-accessible functor which agrees with T iff T is κ-accessible.*

Proof. By definition every small T_κ functor is κ-accessible. Also by definition we see that T_κ agrees on objects with T iff T is κ-accessible.

Hence we only have to prove, that they agree on arrows, i.e given a function $f : X \longrightarrow Y$, the function T_f maps elements in $T_\kappa X$ to elements in $T_\kappa Y$.

In order to see that, let $x \in T_\kappa X$, then there exists $A \subseteq X, |A| < \kappa$ such that $x \in T_i(TA)$ that is there exists $p \in TA$ such that $T_i(p) = x$. Now notice that the diagram

$$\begin{tikzcd} A \arrow[r, "i"] \arrow[d, "f\restriction_A"'] & X \arrow[d, "f"] \\ f(A) \arrow[r, "i'"'] & Y \end{tikzcd}$$

commutes, where the i's are the respective inclusions and $f(A)$ is the direct image of A under f. Then, since T is a functor, we conclude that

$$\begin{tikzcd}
TA \arrow[r, "T_i"] \arrow[d, "T_{f\upharpoonright_A}"'] & TX \arrow[d, "T_f"] \\
Tf(A) \arrow[r, "T_{i'}"'] & TY
\end{tikzcd}$$

also commutes. From this last diagram we have $T_f(x) = T_fT_i(p) = T_{i'}T_f(p)$. Therefore $T_f(x) \in T_i(Tf(A))$, but by definition $|f(A)| < \kappa$, we conclude $T_f(x) \in T_\kappa(Y)$. This concludes the proof. □

Our presentation of Moss' language is based in the following result, the proof can be found in [1].

Theorem 3.1.4. *If T is an accessible functor. Then there exists an initial T-algebra.*

We will only see the use of this result after we have defined Moss' language.

Remark 3.1.5. *There are interesting issues about the relation between the category Coalg(T) and the categories Coalg(T_κ). We claim that the category Coalg(T) is a colimit of the categories Coalg(T_κ). But, as we said before, this is not relevant for this thesis and we will say no more.*

3.1.2 Relation Liftings

We will define a satisfaction relation related to Moss' language using a categorical approach. To explain the set theoretical counter part we will use the following concepts and results:

Theorem 3.1.6 (Epi-mono Factorizations)**.** *In the category Set, every arrow $f : X \longrightarrow Y$ can be factored in the following way:*

$$\begin{tikzcd}
X \arrow[rr, "f"] \arrow[dr, "e"'] & & Y \\
& E \arrow[ur, "m"'] &
\end{tikzcd}$$

where e is a surjection and m is an injection. This factorization is unique up to isomorphisms.

Remark 3.1.7. *The previous result is a particular instance of a more general fact. More explicitly the previous result can be proved in any regular category. Notice that in the category of Set every injection is a monomorphism, also every surjection is an epimorphism, this explain the name of the theorem. What is more interesting is that the arrow e can be constructed to be a regular epimorphism, i.e an epimorphism that is a coequalizer. In the category Set the concepts of surjective, epimorphism, and regular epimorphism are equivalent. The interested reader is refereed to [11] for a general discussion on epi-mono factorizations.*

Using epi-mono factorizations we can define relation liftings.

Definition 3.1.8 (Relation Lifting)**.** *Let $T : Set \longrightarrow Set$ be an endofunctor and let $(\pi_X, \pi_Y) : R \longrightarrow X \times Y$ be a binary relation. The relation lifting of R, written $\overline{T}R$, is the following epi-mono factorization*

$$\begin{tikzcd}
TR \arrow[rr, "{(T(\pi_X), T(\pi_Y))}"] \arrow[dr, "e"'] & & TX \times TY \\
& \overline{T}R \arrow[ur, "m"'] &
\end{tikzcd}.$$

Since we are working in the category Set the relation lifting of R using T is in fact a subset of $TX \times TY$. Because of that in the category Set we can give a more concrete characterization of relation liftings, see [12], in the following way:

Proposition 3.1.9. *Let $T : Set \longrightarrow Set$ be an endofunctor. The relation lifting of a relation $R \subseteq X \times Y$ is the object:*

$$\overline{T}(R) = \{(T(\pi_X)(u), T(\pi_Y)(u)) \mid u \in TR\}$$

Relation liftings do not behave nicely in all cases. In our presentation of Moss' language we will restrict our attention to the following functors.

Definition 3.1.10. *A functor T preserves weak pullbacks if and only if*

$$\overline{T}(R \circ S) = \overline{T}R \circ \overline{T}S$$

for all composable relations R, S. In the particular case of set functors a functor T preserves weak pull backs iff it can be extended to the category of sets with relations and usual composition

Remark 3.1.11. *The name given in the previous definition might seem a bit weird since there is no pullback involved. The two definitions are equivalent see [6] or [9]. In this thesis we don't really have to deal with pullbacks and that is why we have chosen the previous statement as our definition.*

Our presentation of Moss' language needs the following propositions.

Proposition 3.1.12. *Let a functor $T : Set \longrightarrow Set$ and a function $f : A \longrightarrow B$ be given. Assume T preserves weak pullbacks, then*

$$\overline{T}(G_f) = G_{T(f)}.$$

Where $G_f, G_{T(f)}$ are the graphs of the respective functions.

Proof. A complete proof is an easy calculation hence is omitted. A sketch of the proof is: Notice that the graph of an arrow $f : A \longrightarrow B$ is the following pullback

$$\begin{tikzcd}
G_f \arrow[r, "\pi_B"] \arrow[d, "\pi_A"'] & B \arrow[d, "1_B"] \\
A \arrow[r, "f"'] & B
\end{tikzcd}.$$

Therefore, since every functor preserves identities and we assume T preserves weak pullbacks, we conclude the desired equality. □

It is also easy to see that

Proposition 3.1.13. *Let a functor $T : Set \longrightarrow Set$ and a function $f : A \longrightarrow B$ be given. Assume T preserves weak pullbacks, then*

$$\overline{T}(\in_B)\overline{T}(G_f) = \overline{T}(G_{f^{-1}})\overline{T}(\in_A).$$

Where $G_f, G_{T(f)}$ are the graphs of the respective functions.

3.2 Coalgebraic Logic

The coalgebraic language invented by Moss is a local language, hence it has two components. One: A set of formulas, called *Moss' language*. Two: A satisfaction relation, called *Moss' satisfaction* and we write it $\Vdash$. As we said, Moss' language can be uniformly defined for a large class of functors. One of the reasons for this is that Moss' satisfaction is univocally determined by Moss' language, i.e. once we define the set of formulas the satisfaction relation will be obtained in a unique way. Because of this we only refer to Moss' language. In the following pages of this chapter we will assume our functors preserve weak pullbacks.

3.2.1 Moss Language

Rutten in [7] explains that during the development of computer science over the past few decades it has become clear that an abstract description of programming languages is desirable, for example to ensure that a particular program does not depend on the particular representation of the data it operates. Rutten, in the same source, remarks that the data types used by computer scientists are often generated from a given collection of operations. This leads computer scientists to use *initial algebras* to obtain the abstract representation they need. This idea can be used to define the set of formulas for Moss' language. The formulas of Moss' language are an abstract representation of the facts that are expressive in the language.

Remark 3.2.1. *Rutten's idea will become more clear when we present an algebraic representation of coalgebraic modal languages in the next chapter. There we will see that using an operational perspective of the coalgebraic modalities we can represent the formulas using an initial algebra.*

One way to understand Moss' language is to obtain a language that is deeply related to a functor T. Following Rutten, a first approach would be to use the initial algebra for the functor T, but such algebra might not exist. This problem is solved using the small T_κ functor, because it is κ-accessible and then Theorem 3.1.4 guarantees the existence of the initial algebra.

But using only the functor T_κ will lead us to a language where sentences cannot be combined, each formula will be an isolated formula. To solve this problem we use $\mathcal{P} + T$ instead of only T. The power set functor will alow us to do conjunctions of formulas. Again to ensure the existence of the initial algebra, we should use $\mathcal{P}_\kappa + T_\kappa$ instead of $\mathcal{P} + T_\kappa$.

Notice that by the universal property of coproducts there exist two functions:

$$\bigwedge : \mathcal{P}_\kappa(\mathcal{L}_T^\kappa) \longrightarrow \mathcal{L}_T^\kappa; \quad \nabla : T_\kappa \mathcal{L}_T^\kappa \longrightarrow \mathcal{L}_T^\kappa.$$

Using these functions, Moss' language can be defined, see [9], in a categorical context that follows Rutten.

Definition 3.2.2 (Moss Language)**.** *Given a functor $T : Set \longrightarrow Set$ that preserves weak pullbacks. We define κ-*Moss' language*, written $\mathcal{L}_T^\kappa$ to be the carrier of $(\mathcal{L}_T^\kappa, \bigwedge, \nabla)$, where this later structure is an initial algebra for the functor $\mathcal{P}_\kappa + T_\kappa$. The operator ∇ is call the Moss' modality.*

Remark 3.2.3. *The reader familiar with Moss' work will notice that this previous definition is not the definition Moss used in [8]. In that paper Lawrence Moss presented the language as follows:*

*Given a standard functor $T : Set \longrightarrow Set$ that preserves weak pullbacks and regular cardinal κ. We define the κ-*Moss' language*, written $\mathcal{L}_T^\kappa$, to be $\mathcal{L}_T^\kappa = (\mathcal{P}_\kappa + T_\kappa)_*$. That is $\mathcal{L}_T^\kappa$ is the least set such that*

$$\mathcal{L}_T^\kappa = \mathcal{P}_\kappa \mathcal{L}_T^\kappa + T_\kappa \mathcal{L}_T^\kappa.$$

This presentation has some problems. For example using this original approach Moss' language is not definable for all Kripke polynomial functors since the homomorphism functors are not standard. Furthermore Moss' has to extend functors to the category of classes with set continuous functions, this has the disadvantage that Moss' language is then a class and not a set.

Rutten in [7] remarks that the least and greatest fixed points of a monotone function are generalized to the least and greatest points of a functor. That is what Moss' uses in [8] based on the assumption that the functor is standard. Rutten then stress that this last fixed points are suitable described as initial algebras and final coalgebras. Hence our presentations is a generalization of Moss' work.

Remark 3.2.4. *Another presentation of Moss' language from a more set theoretical perspective, see [12], is done using the least fixed point as follows:*

*For $T : Set \longrightarrow Set$ be a standard functor that preserves weak pullbacks. we define κ-*Moss' language*, written $\mathcal{L}_T^\kappa$, to be the least class C such that*

1. *$\bigwedge \Phi \in \mathcal{L}_T^\kappa$ if $\Phi \in \mathcal{P}_\kappa(\mathcal{L}_T^\kappa)$.*
2. *For any $P \in T_\kappa(\mathcal{L}_T^\kappa)$ the formula $\nabla P \in \mathcal{L}_T^\kappa$.*

By the comments above we claim that this presentation still produces the same set of formulas.

3.2.2 Moss' Satisfaction

In [7], Rutten shows how induction can de formulated abstractly using initial algebras. Dually using terminal coalgebras we can formulate coinduction. These two concepts are not really relevant to this thesis, but the representation of Moss language will produce a satisfaction relation by induction. Since $\mathcal{L}^{\kappa}_{T}$ is the carrier of the initial algebra for any other $\mathcal{P}_\kappa + T_\kappa$-algebra with carrier A, there exists a unique morphism

$$\mathcal{L}^{\kappa}_{T} \longrightarrow A$$

The existence of such morphism is the definition by induction discussed by Rutten. In our case, we will show that Moss' satisfaction is defined inductively using this procedure. Since we want Moss' language to be a language for coalgebras we need to give an algebraic representation of coalgebras.

This categorical presentation of Moss' Satisfaction can be found in [9]. More precisely, Moss' satisfaction is an initial arrow. To see that we have to define a functor

$$M : Coalg(T)^{op} \longrightarrow Alg(\mathcal{P} + T),$$

we use M for *Moss functor.* As we will see in later sections and chapters, Moss functors play an important role in defining appropriate semantics and translations between languages. In order to define the Moss' functor we must define a couple of natural transformations. These natural transformations are defined by Pattinson in [9]. We deal with conjunctions first.

Definition 3.2.5. *Given a set A we define $\delta_A : \mathcal{P}_\kappa\mathcal{P}A \longrightarrow \mathcal{P}A$ to be*

$$\delta_A(X) = \bigcap X; \text{ where } X \in \mathcal{P}_\kappa\mathcal{P}A.$$

Remark 3.2.6. *Recall that the actions on objects of the covariant power set functor and of the contravariant power set function are the same, these functors differ on arrows. Now notice that the previous functions define a natural transformation between contravariant functors and not between covariant functors. More formally it is:*

Lemma 3.2.7. *The functions δ_A, as defined above, are the components of a natural transformation $\delta : \mathcal{P}_\kappa\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}$, where $\check{\mathcal{P}}$ is the contravariant power set functor. That is, the following diagram*

$$\begin{tikzcd}
\mathbb{S}et & \mathbb{S}et & \\
A \arrow[d, "f"'] & \mathcal{P}_\kappa\check{\mathcal{P}}A \arrow[r, "\delta_A"] & \check{\mathcal{P}}A \\
B & \mathcal{P}_\kappa\check{\mathcal{P}}B \arrow[u, "\mathcal{P}_\kappa(f^{-1})"] \arrow[r, "\delta_B"'] & \check{\mathcal{P}}B \arrow[u, "f^{-1}"']
\end{tikzcd}$$

commutes for every function $f : A \longrightarrow B$.

Now we take care about Moss' modality.

Definition 3.2.8. *We define $\pi_A : T\check{\mathcal{P}}A \longrightarrow \check{\mathcal{P}}TA$ to be*

$$\pi_A(t) = \{s \in TA \mid (s,t) \in \overline{T}(\epsilon_A)\},$$

where ϵ_A is the membership relation for A. We write π for Pattinson.

The following lemma is a modification of Theorem 4.1.7 in [9] .

Lemma 3.2.9 (Pattinson Transformation)**.** *The functions π_A, as defined above, are the components of a natural transformation $\pi : T\check{\mathcal{P}} \longrightarrow \check{P}T$, the Pattinson transformation, that is the following diagram*

$$\begin{tikzcd}
\mathbb{S}et & & \mathbb{S}et & \\
A \arrow[d, "f"'] & T\check{\mathcal{P}}A \arrow[r, "\pi_A"] & \check{P}TA \\
B & T\check{\mathcal{P}}B \arrow[u, "T(f^{-1})"] \arrow[r, "\pi_B"'] & \check{P}TB \arrow[u, "T(f)^{-1}"']
\end{tikzcd}$$

commutes for every function $f : A \longrightarrow B$.

Proof. We want to prove $T(f)^{-1}\pi_B = \pi_A T(f^{-1})$. Suppose $t \in T\mathcal{P}B$ and $s \in A$, then by definition we have

$$s \in T(f)^{-1}\pi_B(t) \text{ iff } (T_f(s), t) \in \overline{T}(\epsilon_B).$$

By definition of composition of relations,

$$(T_f(s), t) \in \overline{T}(\epsilon_B) \text{ iff } (s, t) \in \overline{T}(\epsilon_B)G_{T(f)}.$$

Since T preserves weak pullbacks it commutes with graphs, see proposition 3.1.12, this equivalence can be extended to

$$(T_f s, t) \in \overline{T}(\epsilon_B) \text{ iff } (s, t) \in \overline{T}(\epsilon_B)\overline{T}(G_f).$$

By definition it preserves composition of relations, then our equivalence now is

$$(s, t) \in \overline{T}(\epsilon_B)\overline{T}(G_f) \text{ iff } (s, t) \in \overline{T}(G_{f^{-1}})\overline{T}(\epsilon_A).$$

Again using the fact that the relation lifting preserves graphs we obtain,

$$\begin{aligned}(s, t) \in \overline{T}(G_{f^{-1}})\overline{T}(\epsilon_A) &\text{ iff } (s, T_{f^{-1}}(t)) \in \overline{T}(\epsilon_A) \\ &\text{ iff } s \in \pi_A T_{f^{-1}}(t).\end{aligned}$$

Hence we conclude

$$s \in f^{-1}\pi_B(t) \text{ iff } s \in \pi_A T_{f^{-1}}(t),$$

from that we obtain the desired equality and this concludes the proof. □

If we have a coalgebra $\alpha : A \longrightarrow TA$ we define π_α to be the following composite

$$\begin{tikzcd}
T\check{\mathcal{P}}A \arrow[rr, "\pi_A"] \arrow[dr, "\pi_\alpha"'] & & \check{\mathcal{P}}TA \arrow[dl, "\alpha^{-1}"] \\
& \check{\mathcal{P}}A &
\end{tikzcd}$$

Using this we now consider the following schema:

$Coalg(T)$	$Alg(\mathcal{P}_\kappa + T)$
$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\alpha"'] & B \arrow[d, "\beta"] \\ TA \arrow[r, "T_f"'] & TB \end{tikzcd}$	$\begin{tikzcd} \mathcal{P}_\kappa\check{\mathcal{P}}A + TA \arrow[d, "{[\delta_A, \pi_\alpha]}"'] & \mathcal{P}_\kappa\check{\mathcal{P}}B + TB \arrow[l, "\mathcal{P}_{f^{-1}} + T_{f^{-1}}"'] \arrow[d, "{[\delta_B, \pi_\beta]}"] \\ \check{\mathcal{P}}A & \check{\mathcal{P}}B \arrow[l, "f^{-1}"] \end{tikzcd}$

Notice that from the commutative diagrams in the lemmas 3.2.7 and 3.2.9 implies that the diagram on the right commutes if and only if the diagram on the left does. This shows that the assignation is functorial. We can define the Moss functor following this idea.

Definition 3.2.10 (Moss Functor)**.** *We define a functor*

$$M : Coalg(T_\kappa)^{op} \longrightarrow Alg(T_\kappa + \mathcal{P}_\kappa)$$

in the following way. On objects: Given a T-coalgebra (A, α), the functor M maps it to

$$[\delta_A, \pi_\alpha] : \mathcal{P}_\kappa\mathcal{P}A + T\mathcal{P}A \longrightarrow \mathcal{P}A.$$

On arrows: The functor M maps a morphism $f : (A, \alpha) \longrightarrow (B, \beta)$ to

$$f^{-1} : (\mathcal{P}B, [\delta_B, \pi_\beta]) \longrightarrow (\mathcal{P}A, [\delta_A, \pi_\alpha])$$

Notice that the image of a coalgebra α is in fact its complex algebra using the Moss' modality, therefore another appropriate name for this functor could also be *Complex algebra functor.*

From now on we will assume our functor is in fact κ-accessible, and we will only stress the subindex in particular important occasions. Using Moss functor we can define Moss' satisfaction as follows.

Definition 3.2.11 (Moss Satisfaction)**.** *Let $T : Set \longrightarrow Set$ be a functor that preserves weak pullbacks and let (A, α) be a T_κ-coalgebra. We define*

$$[-]_\alpha : \mathcal{L}_T^\kappa \longrightarrow \mathcal{P}A$$

to be the unique arrow $[-]_\alpha : (\mathcal{L}_T^\kappa, [\bigwedge, \nabla]) \longrightarrow (\mathcal{P}A, [\delta_A, \pi_A])$ that exists in $Alg(\mathcal{P}_\kappa + T_\kappa)$ because the algebra on the left is initial. We define $\Vdash_\alpha$ in the following way:

$$s \Vdash_\alpha \varphi \textit{ iff } s \in [\varphi]_\alpha.$$

Notation 3.2.12. *We fix some notation before going on. Let $[-]_\alpha : \mathcal{L}_T^\kappa \longrightarrow \check{\mathcal{P}}A$ be Moss' satisfaction for some coalgebra α and let $\varphi \in T\mathcal{L}_T^\kappa$. Instead of writing*

$$T([-]_\alpha)(\varphi)$$

we will write

$$T([\varphi]_\alpha)$$

We will keep the same notation for extensions of Moss' language.

Pattinson transformations are useful because we can compute the satisfaction set of a formula $\nabla\varphi$ as follows.

$$[\nabla\varphi]_\alpha = \alpha^{-1}\pi_A T([\varphi]) = \{s \in A \mid (\alpha(s), T([\varphi])) \in \overline{T}(\in_A)\}.$$

Remark 3.2.13. *Since π_A is defined using relation lifting we can see that Moss satisfaction is also characterized by following clauses, see [12]:*

Let T : Set $\longrightarrow$ Set be a functor that preserves weak pullbacks. Given a T-coalgebra (A, α) we define $\Vdash_\alpha \subseteq A \times \mathcal{L}^\kappa_T$ as the least relation satisfying:

$$s \Vdash_\alpha \bigwedge \varphi \text{ iff } s \Vdash_\alpha \varphi \text{ for all } \varphi \in \Phi$$
$$s \Vdash_\alpha \nabla P \text{ iff } (\alpha(s), P) \in \overline{T}W \text{ for some set } W \subseteq \Vdash_\alpha .$$

Remark 3.2.14. *Before going on we Notice that any algebra for the functor $\mathcal{P}_\kappa + T$ should have an operator corresponding to the operator*

$$\bigwedge : \mathcal{P}_\kappa(\mathcal{L}^\kappa_T) \longrightarrow \mathcal{L}^\kappa_T,$$

and another operator corresponding to the operator

$$\nabla : T_\kappa \mathcal{L}^\kappa_T \longrightarrow \mathcal{L}^\kappa_T.$$

Because of that we will use the same notation on any algebra for $\mathcal{P}_\kappa + T$, i.e an algebra is of the form $(A, [\wedge, \nabla])$. In the particular case where such algebra is in fact the complex algebra of a coalgebra α we write

$$(\check{\mathcal{P}}A, [\wedge_\alpha, \nabla_\alpha]).$$

Notice that ∇_α is equal to $\alpha^{-1}\nabla_A$ where ∇_A is the A-component of Pattinson transformation. At this point this is a purely notational convention but from next section and specially on Chapter 6 it will show to have a much deeper motivation. Namely we are going to translate modalities as algebraic operators instead of modalities as formulas. We will say no more now.

Using Moss functors the following is immediate

Theorem 3.2.15. *The Moss' language associated with a functor T_κ that preserves weak pull backs is adequate with respect to behavioral equivalence.*

It is not that immediate, but also possible to show:

Theorem 3.2.16. *The Moss' language associated with a functor T_κ that preserves weak pull backs has the Hennessy-Milner property with respect to behavioral equivalence.*

Proofs of this fact can be found in [8], Section 7, in [12], Theorem 194, and in [9], Theorem 4.1.9.

3.3 Examples

In the previous section we presented the Moss' language and Moss' satisfaction formally. In this section we will show how Moss' language and Moss' satisfaction relation look like in the case of Kripke polynomial functors. The reader will be refereed back to this section in Chapter 6.

3.3.1 The Identity Functor

A coalgebra for the identity functor is a function $\alpha : A \longrightarrow A$. We compute Moss Logic for the identity functor.

In the case of the identity functor we can apply Moss' modality to any formula. In this case, the relation lifting of a relation is the relation itself, hence the semantic of ∇ is the following

$$s \Vdash_\alpha \nabla\varphi \text{ iff } \alpha(s) \Vdash_\alpha \varphi.$$

The semantics for the boolean operators is defined as usual.

3.3.2 Constant Functors

In this section we assume we are working with a constant functor with value D. A coalgebra for such functor is a function $\alpha : A \longrightarrow D$.

A formula φ in the language $\mathcal{L}_D$ is either an element in D or a boolean combination of elements in D.

The relation lifting of a relation is the equality between elements of D. The semantics for ∇ for a formula $\varphi \in D$ is:

$$s \Vdash_\alpha \nabla\varphi \text{ iff } (\alpha(s), \varphi) \in \overline{D}(\Vdash_\alpha) \text{ iff } \alpha(s) = \varphi.$$

3.3.3 $Hom(D, -)$ Functors

We first recall that the action of a functor $Hom(D, -)$ on arrows is: An arrow $f : A \longrightarrow B$ is mapped to the function

$$f \circ (-) : Hom(D, A) \longrightarrow Hom(D, B).$$

Such function maps an arrow $h : D \longrightarrow A$ to $fh : D \longrightarrow B$.

Notice that a coalgebra for this functor is a function $\alpha : A \longrightarrow Hom(D, A)$, therefore for each state $s \in A$ the coalgebra maps associates a function $\alpha(x) : D \longrightarrow A$.

The Moss language for $Hom(D, -)$ is more complex than the previous ones. The operator ∇ acts on functions $\varphi : D \longrightarrow \mathcal{L}_{Hom(D,-)}$. Given $d \in D$, we will write $\varphi(d)$ for the value of d under the function φ. The relation lifting of the membership relation is

$$\overline{Hom(D, \in_A)} = \{(f, f') \,|\, f(d) \in_A f'(d)) \text{ for all } d \in D\}.$$

Using that we conclude that the semantics for the nabla is

$$s \Vdash_\alpha \nabla\varphi \text{ iff } (\forall d \in D)(\alpha(x)(d) \Vdash_\alpha \varphi(d)).$$

3.3.4 Power set functor

A coalgebra for the covariant power set functor is a function $\alpha : A \longrightarrow \mathcal{P}A$. It is well know that coalgebras for the covariant power set functor are Kripke frames over one relation R, i.e the category of $\mathcal{P}$-coalgebras with coalgebraic morphism is isomorphic to the category of Kripke frames with one relation and bounded morphism.

The nabla operator acts on subsets $\Phi \subseteq \mathcal{L}^{\kappa}_{T}$. The relation lifting of the membership relation with the covariant power set functor is

$$\begin{aligned}\overline{\mathcal{P}}(\in_A) = \{(Q_0, Q_1) \mid (\forall q_0 \in Q_0)(\exists (q_1 \in Q_1))(q_0 \in_A q_1) \\ \text{and } (\forall q_1 \in Q_1)(\exists (q_0 \in Q_0))(q_0 \in_A q_1)\}\,.\end{aligned}$$

Now assume $Q_0 = \alpha(s)$ for some coalgebra and some state s. Also assume Q_1 is the image of set of formulas Φ under the covariant power set functor, i.e it is the set of satisfaction sets of the formulas in Φ. We use the same name Φ to avoid extra notation.

Consider the first part of the formula defining the relation lifting,

$$(\forall q_0 \in Q_0)(\exists q_1 \in Q_1)(q_0 \in_A q_1),$$

using our assumptions above it will become

$$(\forall s' \in \alpha(s))(\exists \varphi \in \Phi)(s' \in [\varphi]).$$

Using the basic modal operators we can see that this is expressing $\Box \bigvee \Phi$.

Now consider the other side of the formula,

$$(\forall q_1 \in Q_1)(\exists (q_0 \in Q_0))((q_0, q_1) \in (\in_A)),$$

Using the conventions above this becomes

$$(\forall \varphi \in \Phi)(\exists s' \in \alpha(s))(s' \in [\varphi]).$$

In the basic modal language it expresses $\bigwedge \Diamond \Phi$. Using this two representations we write Moss' satisfaction for the nabla operator as follows

$$s \Vdash_\alpha \nabla\Phi \text{ iff } s \models_\alpha \Box \bigvee \Phi \wedge \bigwedge \Diamond \Phi,$$

where $\models_\alpha$ is the usual satisfaction relation for Kripke frames.

3.3.5 Nablas as operators

Before going on with examples we will continue the discussion started in remark 3.2.14 on page 26.

To understand Moss' modality is to understand its semantics. As we have seen the semantics of Moss' modalities is given by Pattinson transformations. Recall that Pattinson transformations, on page 25, are natural transformations

$$\nabla : T\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}T,$$

with the following components. An element $t \in T\check{\mathcal{P}}A$ is mapped to

$$\nabla_A(t) = \{s \in TA \,|\, (s,t) \in \overline{T}(\in_A)\}.$$

Using this natural transformation, for each T coalgebra α we obtain a function

$$\nabla_\alpha = \alpha^{-1}\nabla_A : T\check{\mathcal{P}}A \longrightarrow \check{\mathcal{P}}A.$$

As we saw after the definition of Moss' satisfaction giving a formula $\varphi \in T\mathcal{L}^{\kappa}_{T}(\neg)$ the satisfaction set $[\nabla\varphi]_\alpha$ can be computed as follows

$$[\nabla\varphi]_\alpha = \alpha^{-1}\nabla_A T([\varphi]_\alpha).$$

Our conclusion is that in order to understand the semantics of the Moss' modality it is enough to understand the action of the components of the natural transformation

$$\nabla : T\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}T.$$

Now we claim that we can compute Pattinson transformation using the corresponding natural transformations of the ingredients of T. In other words we can understand the Moss' modality of a functor T using the Moss' modalities of its ingredients. Let's consider the case of coproducts to explain our point.

Coproducts of functors

Let $K_1 + K_2$ be the coproduct of two Kripke polynomial functors. We want to describe the natural transformation

$$\nabla : (K_1 + K_2)\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}(K_1 + K_2)$$

in terms of the natural transformations

$$\nabla_1 : K_1\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}K_1$$
$$\nabla_2 : K_2\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}K_2.$$

A first problem is that ∇ is not the coproduct of ∇_1 and ∇_2 because $\check{\mathcal{P}}(K_1 + K_2)$ is not equal, or even isomorphic, to $\check{\mathcal{P}}(K_1) + \check{\mathcal{P}}(K_2)$.

Notice that the relation lifting of the membership relation using a coproduct of functors is

$$\overline{K_1 + K_2}(\in_A) = \{(i_1(s), i_1(t)) \,|\, (s,t) \in \overline{K_1}(\in_A)\} \cup \{(i_2(s), i_2(t)) \,|\, (s,t) \in \overline{K_2}(\in_A)\},$$

where i_1, i_2 are the appropriate coproduct inclusions.

Using this we can express the the natural transformation ∇ as follows. An element $t \in K_1\mathcal{P}A + K_2\check{\mathcal{P}}A$ is mapped to

$$\{i_1(s) \in K_1A + K_2A \,|\, s \in \nabla_{(1,A)}(t)\} \text{ if } t \in K_1\check{\mathcal{P}}A.$$
$$\{i_2(s) \in K_1A + K_2A \,|\, s \in \nabla_{(2,A)}(t)\} \text{ if } t \in K_1\check{\mathcal{P}}A.$$

This shows how to express the natural transformation ∇ in terms of ∇_1 and ∇_2. Explicitly the A-component maps an element $t \in K_1\mathcal{P}A + K_2\mathcal{P}A$ to

$$\nabla_A(t) = \nabla_{(1,A)}(t) \cup \nabla_{(2,A)}(t).$$

Now we will show how to compute the satisfaction set of $\nabla\varphi$. Let $\alpha : A \longrightarrow K_1A + K_2A$ be a coalgebra for the coproduct and let $\varphi \in K_1\mathcal{L} + K_2\mathcal{L}$. From the representation above we conclude that our intuition on equation **??** for the case of the coproduct was accurate but not quite. What we really have is

$$[\nabla\varphi]_{12,\alpha} = \nabla_{1,\alpha}K_1([\varphi]_{12}) \cup \nabla_{2,\alpha}K_2([\varphi]_{12}).$$

In other words this can be express as follows

$$[\nabla\varphi]_{12,\alpha} = \nabla_{1,\alpha}K_1([\varphi]_{12}) \text{ if } \varphi \in K_1\mathcal{L}^{\kappa}_{T}(\neg).$$
$$[\nabla\varphi]_{12,\alpha} = \nabla_{2,\alpha}K_2([\varphi]_{12}) \text{ if } \varphi \in K_2\mathcal{L}^{\kappa}_{T}(\neg),$$

where $T = K_1 + K2$.

A more logical presentation of the satisfaction relation would be

$$s \Vdash_\alpha \nabla\varphi \text{ iff } (\alpha(s) \in K_1A \wedge (s \in \nabla_1K_1[\varphi])) \vee (\alpha(s) \in K_2A \wedge (s \in \nabla_2K_2[\varphi])).$$

Remark 3.3.1. *Another reason why we decided to use Pattinson transformations to describe Moss' modality for coproducts is the following issue. For a formula $\varphi \in K_1\mathcal{L}^{\kappa}_{T}(\neg)$, where $T = K_1 + K_2$, we would be tempted to describe the formula $\nabla\varphi$ saying that it is like the formula $\nabla_1\varphi$, where ∇_1 is Moss' modality for K_1.*

But this does not makes sense because Moss' modality for the coproduct is a function $\nabla : T\mathcal{L}^{\kappa}_{T}(\neg) \longrightarrow \mathcal{L}^{\kappa}_{T}(\neg)$ and Moss' modality ∇_1 is a function $\nabla_1 : K_1\mathcal{L}_{K_1}(\neg) \longrightarrow \mathcal{L}_{K_1}(\neg)$. Hence, we don't know, in principle, if $\varphi \in K_1\mathcal{L}_{K_1}(\neg)$.

If we see that the Moss' modality is in fact a natural transformation we can describe this natural transformation using the ingredients of T, as we saw in the case of the coproduct. Then we solve the issue mentioned in this remark because even that we can not apply ∇_1 to a formula $\varphi \in \mathcal{L}^{\kappa}_{T}(\neg)$, we can apply the natural transformation defining ∇_1 to the satisfaction set $T([\varphi]_{12})$ and then obtain the satisfaction set $[\nabla\varphi]$.

The moral of the history is: We can describe the Pattinson transformation associated with a functor T using the Pattinson transformations of the ingredients of T. We exemplify this idea on the other inductive cases of Kripke polynomial functors.

3.3.6 Product of functors

Let a pair of functors $K_1, K_2 : Set \longrightarrow Set$ be given. In this section we will investigate Moss language and the Coalgebraic modal language for the product functor $K_1 \times K_2 : Set \longrightarrow Set$. We write $[-]_{12}$ for the satisfaction relation, $[-]_1, [-]_2$ for the satisfaction relations of K_1 and K_2, respectively. ∇ is the Moss' modality for the product, and ∇_1, ∇_2 are the modalities for the respective factors.

A coalgebra for a product functor is an arrow $\alpha : A \longrightarrow K_1A \times K_2A$. Hence given $s \in A$, we have $\alpha(s) = (s_1, s_2)$ for some $s_1 \in K_1A$ and $s_2 \in K_2A$. That is a coalgebra over the coproduct is the product of a coalgebra α_1 for K_1 and a coalgebra α_2 for K_2, both with the same carrier.

The nabla operator for a product functor acts on pair of formulas (φ_1, φ_2) such that

$$\varphi_1 \in K_1\mathcal{L} \text{ and } \varphi_2 \in K_2\mathcal{L}.$$

By definition we have

$$s \Vdash_\alpha \nabla(\varphi_1, \varphi_2) \text{ iff } (s_1, s_2), ((K_1 \times K_2)([-])(\varphi_1, \varphi_2)) \in \overline{K_1 \times K_2}(\in_A).$$

The relation lifting for the product is given by

$$\overline{K_1 \times K_2}(\in_A) = \{((x_0, x_1), (x_0', x_1')) \mid (x_0, x_0') \in \overline{K_1}(\in_A) \text{ and } (x_1, x_1') \in \overline{K_2}(\in_A)\}.$$

From that unravelling the definition of satisfaction for a product functor we obtain the following interpretation.

$$s \Vdash_\alpha \nabla(\varphi_1, \varphi_2) \text{ iff } \pi_1\alpha(s) \in \nabla_1 K_1([\varphi_1]_{12} \text{ and } \pi_2\alpha(s) \in \nabla_2 K_2([\varphi_2]_{12}).$$

Again we remark that in this case seeing the modalities as operator is crucial. From this we conclude that the nabla operator for the product acts as follows on complex algebras: Given $t_1 \in K_1\mathcal{P}A$ and $t_2 \in K_2\mathcal{P}A$,

$$\nabla(t_1, t_2) = \nabla_1(t_1) \cap \nabla_2(t_2).$$

3.3.7 $Hom(D, K))$ functors

Now we will compose a Kripke polynomial functor, say K, with a homomorphisms functor, say $Hom(D, -)$. A coalgebra for this functor is a function $\alpha : A \longrightarrow Hom(D, KA)$. Notice that the functor $Hom(D, K(-))$ is a particular case of the product functors, namely multiply T with itself D-times. Hence in the case of finitary Kripke polynomial functors it is just the case studied in the previous section. Anyway we can still analyze on the case of an arbitrary D.

The nabla operator acts on formulas $\varphi \in Hom(D, K\mathcal{L}^\kappa_{Hom(D,K)})$, i.e a formula is a function $\varphi : D \longrightarrow K\mathcal{L}^\kappa_{Hom(D,K)}$. The relation lifting of the membership relation is

$$\overline{Hom(D, K(\in_A))} = \{(f, f') \mid (f(d), f'(d)) \in \overline{K}(\in_A)\}.$$

From that we conclude that the semantics for the nabla operator is

$$s \Vdash_\alpha \nabla\varphi \text{ iff } (\forall d \in D)(\alpha(s)(d) \in \nabla\text{`}K([\varphi(d)])),$$

where ∇' is the nabla operator for K and $[-]$ is the Moss' satisfaction for $Hom(D, K)$.

3.3.8 $\mathcal{P}K$ functors

We finish this section composing the covariant power set functor with a Kripke polynomial functor. A coalgebra for this functor is a function $\alpha : A \longrightarrow \mathcal{P}KA$.

Such coalgebra can be seen as a frame of type K. The nabla operator acts on formulas $\Phi \subseteq K\mathcal{L}_T^\kappa$. The relation lifting of the membership relation is

$$\begin{aligned}\overline{\mathcal{P}}(\in_A) = \big\{(Q_0, Q_1) \,|\, &(\forall q_0 \in Q_0)(\exists (q_1 \in Q_1))((q_0, q_1) \in \overline{K}(\in_A)) \\ &\text{and } (\forall q_1 \in Q_1)(\exists (q_0 \in Q_0))((q_0, q_1) \in \overline{K}(\in_A))\big\}\,.\end{aligned}$$

Therefore the A-component $\nabla_A : \mathcal{P}K\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}\mathcal{P}K$ of Pattinson transformation acts as follows: A set $Q \subseteq K\check{\mathcal{P}}A$ is mapped to

$$\begin{aligned}\nabla_A(Q) = \big\{U \subseteq KA \,|\, &(\forall q \in Q)(\exists u \in U)((q, u) \in \overline{K}(\in_A)) \\ &\text{and } (\forall u \in U)(\exists q \in Q)((q, u) \in \overline{K}(\in_A))\big\}\,.\end{aligned}$$

From these definitions we obtain

$$\begin{aligned}s \Vdash_\alpha \nabla\Phi \text{ iff } &(\forall s' \in \alpha(s))(\exists (\varphi \in \Phi))((s', K([\varphi])) \in \overline{K}(\in_A)) \\ &\text{and } (\forall \varphi \in \Phi)(\exists (s' \in \alpha(s)))((s', K([\varphi])) \in \overline{K}(\in_A)).\end{aligned}$$

3.4 Extensions of Moss' Languages

In the following section we show that more important then Moss' satisfaction are Moss functors. They are the crucial point to define appropriate satisfaction relations over coalgebras. Notice that in the given presentations of Moss language there is no mention to negations neither to disjunctions. This is not a coincidence. In this section we will present extensions of Moss language with negations and disjunctions.

An extension of Moss language can be done using a κ-accessible functor $H : Set \longrightarrow Set$ and a natural transformation

$$\gamma : H\check{P} \longrightarrow \check{\mathcal{P}}.$$

Using γ we define a Moss functor

$$M(\gamma) : Coalg(T) \longrightarrow Alg(\mathcal{P}_\kappa + T + H)$$

in the obvious way. Explicitly: On Arrows $M(\gamma)$ acts like M, i.e $M(\gamma)_f = f^{-1}$. On Objects: The functor $M(\gamma)$ maps a coalgebra (A, α) to the following algebra

$$[\wedge_\alpha, \nabla_\alpha, \gamma_\alpha] : \mathcal{P}\mathcal{P}A + T\mathcal{P}A + H\mathcal{P}A \longrightarrow \mathcal{P}A.$$

The fact that γ is a natural transformation is exactly what we need to prove that $M(\gamma)$ is in indeed a functor.

3.4.1 Moss' Language + Negations

Negation is added using the identity functor, written I, and the complement transformation

$$\neg : \check{P} \longrightarrow \check{P}.$$

We remark two facts: First that a component $\neg_A : \mathcal{P}A \longrightarrow \mathcal{P}A$ of the complement transformation maps a subset $X \subseteq A$ to its complement. Second, the complement transformation is in fact a natural transformation.

Definition 3.4.1 (Moss Language + Negations)**.** *Given a functor $T : Set \longrightarrow Set$ that preserves pullbacks. We define* Moss coalgebraic language with negations, *written $\mathcal{L}^{\kappa}_{T}(\neg)$ to be the carrier of the initial object, say $(\mathcal{L}^{\kappa}_{T}(\neg), [\bigwedge, \nabla, \neg])$, in the category $Alg(\mathcal{P}_{\kappa} + T + I)$.*

We use $M(\neg)$ for the associated Moss functor.

Definition 3.4.2 (Moss Satisfaction for negations)**.** *Let $T : Set \longrightarrow Set$ be a functor that preserves weak pullbacks and let (A, α) be a T-coalgebra. We define*

$$[-]_{\alpha} : \mathcal{L}^{\kappa}_{T}(\neg) \longrightarrow \mathcal{P}A$$

to be the unique arrow $[-]_{\alpha} : (\mathcal{L}^{\kappa}_{T}, [\bigwedge, \nabla, \neg]) \longrightarrow (\mathcal{P}A, [\wedge_{\alpha}, \nabla_{\alpha}, \neg_{\alpha}])$ that exists in $Alg(\mathcal{P}_{\kappa} + T + I)$. We define $\Vdash_{\alpha}$ in the following way:

$$x \Vdash_{\alpha} \varphi \text{ iff } x \in [\varphi]_{\alpha}.$$

The commutativity of the diagram

$$\begin{tikzcd}
\mathcal{L}^{\kappa}_{T}(\neg) \arrow[r, "{[-]_{\alpha}}"] \arrow[d, "\neg"'] & \mathcal{P}A \arrow[d, "\neg_{\alpha}"] \\
\mathcal{L}^{\kappa}_{T}(\neg) \arrow[r, "{[-]_{\alpha}}"'] & \mathcal{P}A
\end{tikzcd}$$

shows that the functor $M(\neg)$ provides the right semantics.

3.4.2 Moss' Language + Disjunctions

We add disjunctions using the multiplication associated to the power set monad, i.e using unions.
A component of this multiplication is a function

$$\mu_{A} : \mathcal{P}_{\kappa}\mathcal{P}A \longrightarrow \mathcal{P}A.$$

such function maps a family of set $X \subseteq \mathcal{P}A$ to its union. i.e

$$\mu_{A}(X) = \bigcup X; \text{ where } X \in \mathcal{P}_{\kappa}\mathcal{P}A.$$

Definition 3.4.3 (Moss Language + Disjunctions)**.** *Given a functor $T : Set \longrightarrow Set$ that preserves pullbacks. We define* Moss coalgebraic language with disjunctions, *written $\mathcal{L}^{\kappa}_{T}(\bigvee)$ to be the carrier of the initial object, say $(\mathcal{L}^{\kappa}_{T}(\neg), [\bigwedge, \nabla, \mu])$, in the category $Alg(\mathcal{P}_{\kappa} + T_{\kappa} + \mathcal{P}_{\kappa})$.*

We use $M(\bigvee)$ for the associated Moss functor.

Definition 3.4.4 (Moss Satisfaction for disjunctions)**.** *Let $T : Set \longrightarrow Set$ be a functor that preserves weak pullbacks and let (A, α) be a T-coalgebra. We define*

$$[-]_{\alpha} : \mathcal{L}^{\kappa}_{T}(\bigvee) \longrightarrow \mathcal{P}A$$

to be the unique arrow $[-]_{\alpha} : (\mathcal{L}^{\kappa}_{T}, [\bigwedge, \nabla, \mu]) \longrightarrow (\mathcal{P}A, [\wedge_{\alpha}, \nabla_{\alpha}, \vee_{\alpha}])$ that exists in $Alg(\mathcal{P}_{\kappa} + T_{\kappa} + \mathcal{P}_{\kappa})$. We define $\Vdash_{\alpha}$ in the following way:

$$x \Vdash_{\alpha} \varphi \text{ iff } x \in [\varphi]_{\alpha}.$$

3.4.3 Moss' Language + Negations + Disjunctions

In principle using the extension procedure described at the beginning of this section we could also add disjunctions to the language $\mathcal{L}^{\kappa}_{T}(\neg)$. But it is well know that disjunctions can be expressed using conjunctions and negations. In this section we will show that this fact has a categorical representation.

The best know translation for disjunctions is

$$\vee = \neg \wedge \neg.$$

Using this translation we will produce a translation functor and an equivalence.

Definition 3.4.5 (Translation Functor)**.** *We define a translation functor,*

$$F_{\vee} : Alg(\mathcal{P}_{\kappa} + T + I) \longrightarrow Alg(\mathcal{P}_{\kappa} + T + \mathcal{P}_{\kappa}),$$

in the following way: The functor $F_{\vee}$ is the identity on arrows. On objects: The functor $F_{\vee}$ maps an algebra $(A, [\wedge_A, \nabla_A, \neg_A])$ to the algebra

$$(A, [\wedge_A, \nabla_A, \neg_A \wedge_A \mathcal{P}_{\neg_A}]).$$

In more details we can describe the translation functor to be the identity on the Moss language and disjunctions are expressed by the following composite:

$$\begin{tikzcd} \mathcal{P}A \arrow[r, "\wedge_A"] & A \arrow[d, "\neg_A"] \\ \mathcal{P}A \arrow[u, "\mathcal{P}_{\neg_A}"] \arrow[r, dashed] & A \end{tikzcd}$$

Proposition 3.4.6. *The translation functor is in fact a functor.*

Proof. It is enough to show that for a morphism $f : A \longrightarrow B$ of $\mathcal{P}_{\kappa} + T + I$ algebras the following diagram

$$\begin{tikzcd} A \arrow[r, "f"] & B \\ \mathcal{P}A \arrow[u, "\neg_A \wedge_A \mathcal{P}_{\neg_A}"] \arrow[r, "\mathcal{P}_f"'] & \mathcal{P}B \arrow[u, "\neg_B \wedge_B \mathcal{P}_{\neg_B}"'] \end{tikzcd}$$

commutes. Since f is a morphism of $\mathcal{P}_{\kappa} + T + I$-algebras the following two diagrams

$$\begin{tikzcd} A \arrow[r, "f"] & B \\ \mathcal{P}A \arrow[u, "\wedge_A"] \arrow[r, "\mathcal{P}_f"'] & \mathcal{P}B \arrow[u, "\wedge_B"'] \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "f"] & B \\ A \arrow[u, "\neg_A"] \arrow[r, "f"'] & B \arrow[u, "\neg_B"'] \end{tikzcd}$$

commute. From that using the power set functor we conclude the commutativity of the first diagram. This concludes the proof. □

Proposition 3.4.7. *The translation functor $F_\vee$ is full and faithful, and disjunctions are expressible in the language $\mathcal{L}^\kappa_T(\neg)$.*

Proof. he first claim follows by construction. The second claim follows from the commutativity of the following diagram

$$\begin{tikzcd}
Alg(\mathcal{P}_\kappa + T + I) \arrow[rr, "F_\vee"] & & Alg(\mathcal{P}_\kappa + T + \mathcal{P}_\kappa) \\
& Coalg(T)^{op} \arrow[ul, "M(\neg)"] \arrow[ur, "M(\vee)"'] &
\end{tikzcd}$$

□

Proposition 3.4.8. *The categories $Alg(\mathcal{P}_\kappa + T + I + \mathcal{P}_\kappa)$ and $Alg(\mathcal{P}_\kappa + T + I)$ are equivalent.*

Proof. Consider the forgetful functor

$$U : Alg(\mathcal{P}_\kappa + T + I + \mathcal{P}_\kappa) \longrightarrow Alg(\mathcal{P}_\kappa + T + I).$$

Clearly the forgetful functor is full and faithful. Now every algebra $(A, [\wedge_A, \nabla_A, \neg_A])$ is isomorphic, in fact equal, to the image of the algebra $(A, [\wedge_A, \nabla_A, \neg_A, \neg_A \wedge_A \mathcal{P}_{\neg_A}])$ under the functor U. Therefore the functor U is essentially surjective on objects, hence is an equivalence. This concludes the proof. □

Notice that the forgetful functor is not injective in objects therefore is not an isomorphism.

Chapter 4

From Modal Logic to Coalgebraic Modal Logic via Yoneda Lemma

In the previous chapter, we studied the first language developed to express facts about coalgebras that could be defined uniformly for a large class of functors. As we saw in the case of the power set functor, section 3.3.4, the basic modalities are inside the nabla, see equation 3.3.4 on page 28. Furthermore in the case of the power set functor, we can recover the basic modalities from the nabla operator. We can easily see that

$$\Diamond\varphi = \nabla\{\varphi, \top\}; \text{ and } \Box\varphi = \nabla\emptyset \vee \nabla\{\varphi\}.$$

Now we would like to change the functor but still obtain modalities from the nabla operator. We want to obtain modalities that are like the basic modalities $\Diamond, \Box$ but for arbitrary functors. In this chapter we don't show how to obtain such modalities. A non constructive procedure can be found on Chapter 5, another more constructive answer for the case of Kripke polynomial functors can be found on Chapter 6.

In this chapter we will show how to generalize the basic modalities to arbitrary functors. One of the key points is to see that coalgebraic modalities, as nabla operators, are rather operators over sets instead of just formulas, i.e. they can be described using natural transformations. Using this perspective we will present the basic modalities as natural transformations. With this we will define the concept of *predicate lifting* that we claim is a good generalization of the concept of modality.

To put it in another way,

Using predicate liftings each coalgebra induces a neighborhood model.

In other words, using predicate liftings we can transform an obscure, and not really known, T-coalgebra into a more familiar structure like a neighborhood model. This other intuition will show to be the right approach to present coalgebraic modal languages algebraically, i.e. as the carrier of some initial algebra, see

Section 4.5. We will not discus this approach right now. Instead we will begin studying the basic modalities of the basic modal language to present predicate liftings as a generalization of the basic modalities.

4.1 Basic Modal Logic Revisited

First we analyze the case of the universal modality. Recall that a Kripke frame is a coalgebra for the power set functor. Now, given a formula of the basic modal language φ, what can we say about the formula

$$\Box\varphi.$$

This last formula has a well known semantics using the relational approach. We will restate this definition using the coalgebraic perspective. Let a Kripke frame $\alpha : A \longrightarrow \mathcal{P}A$ and a state $s \in A$ be given. Translating the relational semantics for the box, we obtain:

$$s \models_\alpha \Box\varphi \text{ iff } (\forall s' \in \alpha(s))(s' \models_\alpha \varphi).$$

Restating this using satisfaction sets we obtain

$$s \in [\Box\varphi] \text{ iff } \alpha(s) \subseteq [\varphi]. \tag{4.1}$$

Now notice two facts: first, an equivalence like the previous one is associated with every formula φ. Second, we are transforming a membership relation into a subset relation in the same way the power set functor does. Then we can see the box like a transformation of satisfaction sets. Consider the following function

$$\Box_A : 2^A \longrightarrow 2^{\mathcal{P}A}$$

defined as follows

$$\Box_A(X) = \{Y \subseteq A \,|\, Y \subseteq X\}.$$

Using this, function the semantics for the universal modality can be expressed as follows

$$s \models_\alpha \Box\varphi \text{ iff } \alpha(s) \in \Box_A([\varphi]).$$

Now notice that if $f : A \longrightarrow B$ is a function, the following diagram

$$\begin{tikzcd}
\mathbb{S}et & \mathbb{S}et & \\
A \arrow[d, "f"'] & 2^A \arrow[r, "\Box_A"] & 2^{\mathcal{P}A} \\
B & 2^B \arrow[u, "f^{-1}"] \arrow[r, "\Box_B"'] & 2^{\mathcal{P}B} \arrow[u, "(\mathcal{P}(f))^{-1}"']
\end{tikzcd}$$

commutes. That is, the universal modality is a natural transformation $\Box : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}\mathcal{P}$.

Using the same kind of analysis we can see that the existential modality is also a natural transformation $\Diamond : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}\mathcal{P}$. This natural transformation has the following components:

$$\Diamond_A(X) = \{Y \subseteq A \,|\, Y \cap X \neq \emptyset\}.$$

Based on this analysis, a natural way (not the only one) to generalize the basic modal operators is to change the functor and keep the naturality condition. Such natural transformations are called *predicate liftings.* Formally predicate liftings are defined as follows:

Definition 4.1.1 (Predicate Liftings)**.** *A* predicate lifting *for a functor T is a natural transformation*

$$\lambda : 2^{(-)} \longrightarrow 2^{T(-)},$$

where $2^{(-)} : Set^{op} \longrightarrow Set$ denotes the contravariant power set functor.

Remark 4.1.2. *The naturality condition happens to be the right condition to obtain adequate languages with respect to behavioral equivalence. In our presentation it will be used to define Schröder functors that are the analogous of Moss' functor for the case of coalgebraic modal languages.*

4.2 Coalgebraic Modal Languages

Now we will present how to recover a language from predicate liftings. We first present coalgebraic modal languages as they are defined in [10]. Later we will present coalgebraic modal languages in a more categorical light, i.e. we will define coalgebraic modal languages as the carrier of initial algebras for an appropriate functor. With this algebraic characterization, we will define the equivalent of Moss' functors for the case of coalgebraic modal languages. We call these Schröder functors.

Our task now is to interpret predicate liftings over coalgebras. As we saw in the previous section the semantics for the universal modality can be restated as follows:

$$s \models_\alpha \Box\varphi \text{ iff } \alpha(s) \in \Box_A([\varphi]),$$

where $\Box_A$ is the A-component of the natural transformation associated with the universal modality. From this it is clear how to define the semantics for a predicate lifting. The idea is that for each predicate lifting, say λ, we will have a symbol in the language. We use $[\lambda]$ to represent representing the modality associated with λ. Using this we define the semantics for λ as follows:

$$s \models_\alpha [\lambda]\varphi \text{ iff } \alpha(s) \in \lambda_A([\varphi]),$$

where λ_A is the A-component of λ. At this point, we have enough machinery to define a coalgebraic modal language. Before doing this, however, we will first discuses polyadic modalities. This because monadic predicate liftings can not handle all the modalities known from modal logic.

Consider the functor $\mathcal{P}((-) \times (-))$. A coalgebra for this functor is a function

$$\alpha : A \longrightarrow \mathcal{P}(A \times A).$$

This coalgebra is in fact isomorphic to a Kripke frame where the successor of a state s is not a single state s' but a pair of states (s', s''). An intuitive modality for α would be a modality that sees the behavior of s and s' at the same time. Notice that predicate liftings cannot, in principle, cover this intuition. Predicate liftings act over a single formula and then only see a successor of an state s as a single object. Therefore we have to switch from predicate liftings to *polyadic predicate liftings.*

Definition 4.2.1 (Polyadic Predicate Liftings). *Let η be a cardinal number. An η-*ary predicate lifting *for a functor T is a natural transformation*

$$\lambda : (2^{(-)})^{\eta} \longrightarrow 2^{T(-)},$$

*where $2^{(-)} : Set^{op} \longrightarrow Set$ denotes the contravariant power set functor. A set Λ of such predicate liftings is called σ-*bounded *for a regular cardinal σ or $\sigma = 1$ if all predicate liftings in Λ have arity smaller than σ.*

The naturality condition says that for all sets A, B and functions $f : A \longrightarrow B$ the following diagram

$$\begin{tikzcd}
\mathbb{S}et & & \mathbb{S}et \\
A \arrow[d, "f"'] & (2^A)^{\eta} \arrow[r, "\lambda_A"] & 2^{TA} \\
B & (2^B)^{\eta} \arrow[u, "(f^{-1})^{\eta}"] \arrow[r, "\lambda_B"'] & 2^{TB} \arrow[u, "(T(f))^{-1}"']
\end{tikzcd}$$

commutes, where λ_A, λ_B are the components of λ and the vertical arrow on the left of the square is the η-fold product of f^{-1}. The commutativity of this diagram is equivalent to the fact that for each family $\overrightarrow{X} = (X_j)_{j\in\eta}$ of subsets $X_j \subseteq B$,

$$(T_f)^{-1}\lambda_B(\overrightarrow{X}) = \lambda_A\overrightarrow{f^{-1}(X)}$$

holds, where $\overrightarrow{f^{-1}(X)} = (f^{-1}(X_j))_{j\in\eta}$.

For example consider the following polyadic predicate lifting for a functor $Hom(D, -)$. We define ν to be the following $|D|$-ary predicate lifting:

$$\begin{aligned}
\nu_A : (2^A)^D &\longrightarrow 2^{Hom(D,A)} \\
(X_d)_{d\in D} &\longrightarrow \{h : D \longrightarrow A \,|\, (\forall d \in D)(h(D) \subseteq X_d)\}.
\end{aligned}$$

It is easy to see that these functions are in fact the components of a natural transformation $\nu : (2^{(-)})^D \longrightarrow 2^{Hom(D,-)}$.

The intuition behind polyadic predicate liftings is that of polyadic modalities.

Now we can define coalgebraic modal languages. This language is parameterized by a set of predicate liftings Λ.

Definition 4.2.2 (Coalgebraic Modal Languages I)**.** *Let $T : Set \longrightarrow Set$ be a κ-accessible functor and let Λ be a set of predicate liftings, possibly polyadic, for T. Then $\mathcal{L}_T^\kappa(\Lambda)$, the κ-*coalgebraic modal language for T*, is defined as the least class such that*

1. *If $\Phi \in \mathcal{P}_\kappa \mathcal{L}_T^\kappa(\Lambda)$ then $\bigwedge \Phi \in \mathcal{L}_T^\kappa(\Lambda)$.*
2. *If $\varphi \in \mathcal{L}_T^\kappa(\Lambda)$ then $\neg\varphi \in \mathcal{L}_T^\kappa(\Lambda)$.*
3. *If λ is a η-ary predicate lifting in Λ and $(\varphi_j)_{j\in\eta}$ is a family of formulas in $\mathcal{L}_T^\kappa(\Lambda)$ then $[\lambda](\varphi_j)_{j\in\eta} \in \mathcal{L}_T^\kappa(\Lambda)$.*

The satisfaction relation is defined as usual.

Definition 4.2.3 (Coalgebraic satisfaction I)**.** *We define the satisfaction relation for conjunctions and negations in the usual way. The definition for a modal operator $[\lambda]$ is:*

$$(\alpha, s) \models [\lambda](\varphi_j)_{j\in\eta} \text{ iff } \alpha(s) \in \lambda_A(\overrightarrow{[\varphi]}),$$

where $\overrightarrow{[\varphi]} = ([\varphi_j])_{j\in\eta}$.

It is our we claim that coalgebraic modal languages are an appropriate generalization of the basic modal language.

4.3 Counting Predicate Liftings

There might seem that there are too many predicate liftings for a functor T, but in fact this is not the case. Lutz Schrdoer in [10] showed that the predicate liftings of arity η are in bijection with the subsets of $T(2^\eta)$. In order to see that we have to use one of the basic results of category theory, namely Yoneda Lemma. The general formulation of Yoneda Lemma is as follows.

Lemma 4.3.1 (Yoneda Lemma)**.** *Let $\mathcal{C}$ be a locally small category, Let $F : \mathcal{C}^{op} \longrightarrow Set$ be a functor. For every object C of $\mathcal{C}$, there is a natural bijection*

$$y_{C,F} : Nat(Hom(-, C), F) \longrightarrow F(C).$$

This bijection is natural in the following sense: Given $g : C' \longrightarrow C$ in $\mathcal{C}$ and $\mu : F \longrightarrow F'$ in $Set^{\mathcal{C}^{op}}$, the diagram

$$\begin{tikzcd}
Nat(Hom(-,C),F) \arrow[r, "y_{C,F}"] \arrow[d, "Nat_{(g,\mu)}"'] & F(C) \arrow[d, "\mu_{C'}F_g = F'_g\mu_C"] \\
Nat(Hom(-,C'),F') \arrow[r, "y_{C',F'}"'] & F'(C')
\end{tikzcd}$$

commutes in the category Set.

Proof. We will only define the natural bijection. A complete proof can be found in [11]. A natural transformation $\lambda : Hom(-, C) \longrightarrow F$ is completely determined by $\lambda_C(1_C) \in F(C)$. Conversely given an element $a \in F(C)$ we define a natural transformation $\lambda_a : Hom(-, C) \longrightarrow F$ such that the component $\lambda_{a,X}$ maps an arrow $f : X \longrightarrow C$ to $\lambda_{X,a}(f) = F_f(a)$. These maps are inverse of each other and satisfy the mentioned naturality condition. $\square$

Notice that given a functor $T : Set \longrightarrow Set$ there is a functor $T^{op} : Set^{op} \longrightarrow Set^{op}$ that acts exactly like T but has different domain and codomain, hence is not equal. Since the actions are the same we use the same name for both functors. Now the functor T^{op} can be composed with any $Hom(-, C)$ functor. We obtain a contravariant functor

$$Hom(T(-), C) : Set^{op} \longrightarrow Set.$$

Then we can restate Yoneda Lemma for such particular functors.

Proposition 4.3.2. *Let $T : Set \longrightarrow Set$ be a functor. For every set C there is a natural bijection*

$$y_{C,T} : Nat(Hom(-, C), Hom(T(-), C)) \longrightarrow Hom(T(C), C).$$

This will also hold in the particular case of $C = 2$, a set with two elements. There is a natural bijection

$$y_{C,T} : Nat(Hom(-, 2), Hom(T(-), 2) \longrightarrow Hom(T(2), 2).$$

Then we conclude that for predicate liftings we have

Corollary 4.3.3. *Let $T : Set \longrightarrow Set$ be a functor. The predicate liftings for T are in natural bijection with the subsets of $T(2)$.*

We will explain explicitly how to obtain a predicate lifting from a set $C \subseteq T(2)$. First notice that such set is in fact a function

$$\chi_C : T(2) \longrightarrow 2.$$

We want to define a predicate lifting, i.e a natural transformation,

$$\lambda_C : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}T.$$

We only explain how to define the components. Fix a set A. The component of λ_C on A, written $\lambda_{C,A}$, according to Yoneda Lemma is defined as follows.

Let X be a subset of A. Since T is covariant we have a function

$$T(\chi_X) : TA \longrightarrow T2.$$

Using the contravariant power set functor we obtain a function

$$(T(\chi_X))^{-1} : \check{\mathcal{P}}T2 \longrightarrow \check{\mathcal{P}}TA.$$

Now the proof of Yoneda Lemma says that in order to obtain the action of $\lambda_{C,A}$ over X, this last function should be evaluated on C , i.e

$$\lambda_{C,A}(X) = (T(\chi_X))^{-1}(C).$$

In an even more set theoretical presentation this set is defined as follows.

$$\lambda_{C,A}(X) = \{t \in TA \,|\, T(\chi_X)(t) \in C\} \tag{4.2}$$

We can obtain a similar representation of polyadic predicate liftings. Notice the following.

Proposition 4.3.4. *For any cardinal number η, the functor $Hom(-,2)^{\eta} : Set^{op} \longrightarrow Set$ is isomorphic to $Hom(-,2^{\eta}) : Set^{op} \longrightarrow Set$.*

Proof. We define a natural isomorphism having as components the following functions: $\mu_X : Hom(X,2)^{\eta} \longrightarrow Hom(X,2^{\eta})$ maps a family of functions $(f_j : X \longrightarrow 2)_{j\in\eta}$ and assigns the product arrow $(f_j)_{j\in\eta}$. Its inverse maps an arrow $f : X \longrightarrow 2^{\eta}$ the family $\pi_j f : X \longrightarrow 2$, where π_j is the j-th component. □

Based on this proposition from now on, in order to simplify computations we will understand polyadic predicate liftings as transformations

$$\lambda : Hom(-,2^{\eta}) \longrightarrow Hom(T(-),2).$$

The previous proposition has as corollary the desired representation.

Corollary 4.3.5. *Let $T : Set \longrightarrow Set$ be a functor and η a cardinal number. The η-ary predicate liftings for T are in natural bijection with the subsets of $T(2^{\eta})$.*

Explicitly, the η-ary predicate lifting associated with a set $C \subseteq T(2^{\eta})$ is computed as follows. Let $(X_j)_{j\in\eta}$ be a family of subsets of A. The family $(X_j)_{j\in\eta}$ defines an arrow

$$< \chi_{X_j} >_{j\in\eta}: A \longrightarrow 2^{\eta},$$

namely the product arrow. Using the functor T over this arrow we obtain

$$T(< \chi_{X_j} >_{j\in\eta}) : TA \longrightarrow T(2^{\eta}).$$

Using the contravariant power set functor we obtain a function

$$((T(< \chi_{X_j} >_{j\in\eta}))^{-1}) : \check{\mathcal{P}}T(2^{\eta}) \longrightarrow \check{\mathcal{P}}TA.$$

Notice that we are using the inverse image of the function $T(< X_j >_{j\in\eta})$. Evaluating this function on C we obtain the action of $\lambda_{C,A}$ over X, i.e

$$\lambda_{C,A}(X) = ((T(< \chi_{X_j} >_{j\in\eta}))^{-1})(C).$$

In a set theoretical presentation this set is defined as follows.

$$\lambda_{C,A}(X) = \{t \in TA \,|\, (T(< \chi_{X_j} >_{j\in\eta})(t) \in C)\}. \tag{4.3}$$

4.3.1 Singleton Predicate Liftings

In this section we will study some important cases of predicate liftings for a functor T. First, the predicate liftings associated with the empty set. Second, the predicate liftings associated with the full sets $T(2^{\eta})$ and third, the predicate liftings associated with singleton sets.

Every functor has at least two predicate liftings of arity η, one associated with the empty set,an another one associated with full set $T(2^{\eta})$. Let us compute first the predicate lifting associated with the empty set. Equation (4.3) describes the predicate lifting as follows

$$\lambda_{\emptyset,A}(X) = \{t \in TA \,|\, T(< \chi_{X_j} >_{j\in\eta})(t) \in \emptyset\} = \emptyset.$$

Notice that there are no coalgebra α, no state $s \in A$ and no formula φ such that

$$s \models_\alpha [\lambda_\emptyset]\varphi$$

holds. It would be possible if our states accepts contradictions. We write this predicate lifting $\lambda_\perp$ instead of $\lambda_\emptyset$. In other words $[\lambda_\perp]\varphi = \perp$. These predicates liftings will play an important role in the translations of coproducts.

Now we compute the predicate lifting associated with the full set $T(2^\eta)$. By equation (4.3) the action of this predicate lifting is

$$\lambda_{T(2^\eta),A}(X) = \{t \in TA \,|\, (T(< \chi_{X_j} >_{j\in\eta})(t) \in 2^\eta)\} = TA.$$

Notice that for every coalgebra α, every state $s \in A$ and every formula φ the following

$$s \models_\alpha [\lambda_{T(2^\eta)}]\varphi$$

holds. We write this predicate lifting $\lambda_\top$ instead of λ_{2^η}. These predicate lifting will play an important role in the translation of products.

One of most basic facts of set theory, but one of the most fundamental ones, is the following: For any set C the following

$$C = \bigcup_{p\in C} \{p\}$$

holds. From that we have that the predicate liftings associated with singleton sets have an important property, namely they generate all other predicate liftings. Let $p \in T(2^\eta)$. Equation (4.3) implies that the predicate lifting associated with $\{p\}$ is

$$\begin{aligned}\lambda_{\{p\},A}(X) &= \{t \in TA \,|\, (T([\chi_{X_j}]_{j\in\eta})(t) \in \{p\})\}\\ &= \{t \in TA \,|\, (T([\chi_{X_j}]_{j\in\eta})(t) = p)\}\end{aligned}$$

From that we can prove the following simple but useful result

Theorem 4.3.6. *Let a functor $T : Set \longrightarrow Set$ and a set $C \subseteq T(2^\eta)$ be given. Then for every set A and every family of subsets $\overrightarrow{X} = (X_j)_{j\in\eta}$ of A, the following*

$$\lambda_{C,A}(\overrightarrow{X}) = \bigcup_{p\in C} \lambda_{\{p,A\}}(\overrightarrow{X})$$

holds.

As corollary we have the particular case of $\eta = 1$.

Corollary 4.3.7. *Let a functor $T : Set \longrightarrow Set$ and a set $C \subseteq T(2)$ be given. Then for every set A and every subset $X \subseteq A$, the following*

$$\lambda_{C,A}(X) = \bigcup_{p\in C} \lambda_{\{p\},A}(X)$$

holds

This is particularly important because now we have the following representation of coalgebraic modalities.

Corollary 4.3.8. *Let $T : Set \longrightarrow Set$ be a functor and s a state in some T-coalgebra α, then for every set $C \subseteq T(2^\eta)$ and every family of formulas $(\varphi_j)_{j\in\eta}$,*

$$s \models_\alpha [\lambda_C](\varphi)_{j\in\eta} \text{ iff } s \models_\alpha \bigvee_{p\in C} [\lambda_{\{p\}}](\varphi)_{j\in\eta}.$$

In the particular case of $\eta = 1$ this is.

Corollary 4.3.9. *Let $T : Set \longrightarrow Set$ be a functor and s a state in some T-coalgebra α, then for every set $C \subseteq T(2)$ and every formula φ*

$$s \models_\alpha [\lambda_C]\varphi \text{ iff } s \models_\alpha \bigvee_{p\in C} [\lambda_{\{p\}}]\varphi.$$

Since such predicate liftings will be of particular importance we give them a name.

Definition 4.3.10 (Singleton Predicate Liftings)**.** *A predicate lifting λ_C associated with a set $C \subseteq T(2^\eta)$ is called a* singleton predicate lifting*, or a* singleton lifting*, if $C = \{p\}$ for some $p \in T(2^\eta)$. We usually write λ_p instead of $\lambda_{\{p\}}$ for the singleton lifting associated with $\{p\}$.*

In Chapter 6 we will use singleton liftings to simplify our work. This last two corollaries show that it will be enough to translate singleton liftings in the case the target language has disjunctions. This will be particularity useful to cover the inductive steps.

4.4 Examples

Until here we have presented coalgebraic modal languages formally and we have defined the concept of singleton lifting. In this section we will show how do the predicate liftings look like in the case of Kripke polynomial functors. Based on Theorem 4.3.6 in most of the cases we will only show how singleton predicate liftings look like. The reader will be refereed back to this section in Chapter 6.

4.4.1 The Identity Functor

A coalgebra for the identity functor is a function $\alpha : A \longrightarrow A$.

According to the first corollary of proposition 4.3.2, the identity functor has four monadic predicate liftings.

From now on during this thesis we will represent the set 2 as follows $2 = \{\bot, \top\}$.

Using this notation the four predicate liftings associated for the identity functor are associated with the following sets

$$\emptyset, \{\bot\}, \{\top\}, \{\bot, \top\}.$$

We will compute explicitly the predicate lifting associated with $\{\top\}$. Equation 4.2 tell us that the A component of this predicate lifting is

$$\lambda_{\{\top\},A}(X) = \{t \in TA \,|\, (\chi_X)(t) \in \{\top\}\} = X.$$

In other words the predicate lifting associated with $\{\top\}$ is the identity: $1 : 2^A \longrightarrow 2^A$. The semantics for this predicate lifting is:

$$s \models_\alpha [1]\varphi \text{ iff } \alpha(s) \models_\alpha \varphi.$$

We can easily see that the other predicate liftings are:

1. The complement: $\neg : 2^A \longrightarrow 2^A$, this corresponds to $\{\bot\}$. The semantics for this predicate lifting is:

$$s \models_\alpha [\neg]\varphi \text{ iff } \alpha(s) \models_\alpha \neg\varphi.$$

2. The top constant: $M : 2^A \longrightarrow 2^A$, this corresponds to $\{\bot, \top\}$. The semantics for this predicate lifting is:

$$s \models_\alpha [M]\varphi \text{ iff } \alpha(s) \models_\alpha \top.$$

3. The bot constant: $\emptyset : 2^A \longrightarrow 2^A$, this corresponds to $\emptyset$. The semantics for this predicate lifting is:

$$s \models_\alpha [\emptyset]\varphi \text{ iff } \alpha(s) \models_\alpha \bot.$$

Polyadic predicate liftings are sequences of the singleton liftings.

4.4.2 Constant functors

In this section we assume we are working with a constant functor with value D. A coalgebra for such functor is a function $\alpha : A \longrightarrow D$.

As a consequence of proposition 4.3.2 we know that D has $2^{|D|}$ predicate liftings. Let $C \subseteq D$, the component, of the predicate lifting λ_C,

$$\lambda_{C,A} : 2^A \longrightarrow 2^D$$

is the constant function with value C. The semantics is given by

$$s \models_\alpha [\lambda_C]\varphi \text{ iff } \alpha(s) \in \lambda_C([\varphi]) \text{ iff } \alpha(s) \in C.$$

In the particular case of singleton sets it is

$$s \models_\alpha [\lambda_p]\varphi \text{ iff } \alpha(s) = p.$$

In this case there is no difference between polyadic predicate liftings an unary liftings.

4.4.3 $Hom(D, -)$ Functors

Notice that a coalgebra for this functor is a function $\alpha : A \longrightarrow Hom(D, A)$, therefore for each state $s \in A$ the coalgebra maps associates a function $\alpha(s) : D \longrightarrow A$.

In this case we only study singleton liftings, i.e we only consider the case $\{P\} \subseteq Hom(D, 2)$, i.e $P \subseteq D$. Since $Hom(D, -)(\chi_X)$ is the function that composes χ_X to the right, we can see that the predicate lifting associated with $\{P\}$ acts as follows: Given $X \subseteq A$

$$\lambda_{P,A}(X) = \{h : D \longrightarrow A \,|\, \chi_X h = \chi_P\}.$$

More generally, given $\{P\} \subset Hom(D, 2^\eta)$, the polyadic predicate lifting associated with $\{P\}$ acts as follows: First notice that P can be seen as a family of subsets of D, say $(P_j)_{j\in\eta}$, then given a family $(X_j)_{j\in\alpha}$

$$\lambda_{P,A}(X_j)_{j\in\eta} = \bigcap_{j\in\eta} \lambda_{P_j,A}(X_j).$$

In this last formula we are using the fact that composition to the right distributes over product arrows.

4.4.4 Power set functor

According to Proposition 4.3.2, the power set functor has 16 predicate liftings. The universal modality correspond to the set $\{\emptyset, \{\top\}\}$. The existential modality correspond to the set $\{\{\top\}, \{\top, \bot\}\}$. Furthermore it can be shown that all other predicate liftings are the boolean combinations of the last two. We only show this characterization for the singleton liftings.

The singleton liftings are the associated with the elements of $\mathcal{P}2$

$$\{\top\},\ \{\bot\},\ \{\bot, \top\},\ \emptyset.$$

Here are they respective semantics. Fix a set A and a subset $X \subseteq A$.

We will first compute the predicate associated with $\{\top\}$. Equation 4.2 tells that the components of this natural transformation are

$$\lambda_A(X) = \{U \subseteq A \,|\, \mathcal{P}(\chi_X)(U) = \top\}$$

Since $\mathcal{P}(\chi_X)$ is the direct image we have that the components of $\lambda_{\{\top\}}$ are

$$\lambda_A(X) = \{U \subseteq A \,|\, U \neq \emptyset \wedge U \subseteq X\}.$$

The semantics for this predicate lifting is given by the formula $\Box\varphi \wedge \Diamond\varphi$.

Using the same procedure we can see that the components of $\lambda_{\{\bot\}}$ are

$$\lambda_A(X) = \{U \subseteq A \,|\, U \neq \emptyset \wedge U \subseteq \neg X\},$$

where $\neg X$ is the complement of X. The semantics for this predicate lifting is given by the formula $\Box\neg\varphi \wedge \Diamond\neg\varphi$.

We will give more details in section 4.4.8 but we can see that the predicate lifting $\lambda_{\{\bot,\top\}}$ has the following components

$$\lambda_A(X) = \{U \subseteq A \,|\, U \cap X \neq \emptyset \neq U \cap \neg X\}.$$

The semantics is given by the formula $\Diamond\varphi \wedge \Diamond\neg\varphi$.

Finally the predicate lifting $\lambda_\emptyset$ is defines as follows

$$\lambda_A(X) = \{\emptyset\}.$$

A state validates this predicate lifting iff it has no successors, i.e validates $\Box\bot$. Notice that the empty set defining this predicate lifting is an element of $\mathcal{P}2$ and not a subset of it.

4.4.5 Coproducts of functors

Let $K_1 + K_2$ be the coproduct of two Kripke polynomial functors. It is well know that $X^{Y+Z} \cong X^Y \times X^Z$. Therefore a predicate lifting

$$\lambda : 2^{(-)} \longrightarrow 2^{(K_1+K_2)(-)} \cong 2^{K_1(-)} \times 2^{K_2(-)},$$

is in fact a pair of predicate liftings

$$\lambda_1 : 2^{(-)} \longrightarrow 2^{K_1(-)} \quad \lambda_2 : 2^{(-)} \longrightarrow 2^{K_2(-)}.$$

Assume $\lambda = (\lambda_1, \lambda_2)$, where λ_1 and λ_2 are predicate liftings for K_1 and K_2 respectively. It is easy to see that for a set $X \subseteq A$,

$$\lambda_A(X) = \lambda_{1,A}(X) \cup \lambda_{2,A}(X).$$

The same holds for polyadic predicate liftings.

Now we characterize singleton liftings for the coproduct. An element $p \in K_1(2^\eta) + K_2(2^\eta)$ belongs to exactly one of the factors. Assume $p \in K_1(2^\eta)$. The predicate lifting for the coproduct associated with $\{p\}$ is $(\lambda_p, \lambda_\bot)$, where $\lambda_\bot$ is the predicate lifting defined on page 43 and λ_p is the predicate lifting for K_1 associated with $\{p\}$. This predicate lifting acts as follows.

$$(\lambda_p, \lambda_\bot)_A(\overrightarrow{X}) = \lambda_p(\overrightarrow{X}),$$

where $\overrightarrow{X} = (X_j)_{j\in\eta}$ Symmetrically if $p \in K_2(2^\eta)$ the predicate lifting for the coproduct associated with $\{p\}$ is $(\lambda_\bot, \lambda_p)$.

A coalgebra for the coproduct is a function $\alpha : A \longrightarrow K_1A + K_2A$, i.e for each state $s \in A$ either $\alpha(s) \in K_1A$ or either $\alpha(s) \in K_2A$. The semantics for a singleton lifting associated with $p \in K_1(2^\eta)$ is

$$s \models_\alpha [\lambda_p, \lambda_\bot](\varphi_j)_{j\in\eta} \text{ iff } \alpha(s) \in \lambda_p([\varphi_j])_{j\in\eta}.$$

Notice that this previous formula implies $\alpha(s) \in K_1A$. The semantics for $p \in K_2(2^\eta)$ is symmetric.

4.4.6 Product of functors

Let a pair of functors $K_1, K_2 : Set \longrightarrow Set$ be given. In this section we will investigate the Coalgebraic modal language for the product functor $K_1 \times K_2 : Set \longrightarrow Set$.

We first studied singleton liftings to understand predicate liftings for the product. The following paragraphs will show that singleton liftings enormously simplify the presentation of predicate liftings for the product of functors.

Let $(p_1, p_2) \in K_1(2^\eta) \times K_2(2^\eta)$. According to equation 4.3, the predicate lifting for the product associated with $\{(p_1, p_2)\}$ acts as follows: Given a family of subsets $(X_j)_{j\in\eta}$ of A

$$\lambda_{(p_1,p_2),A}(X) = \{(t_1, t_2) \in K_1A \times K_2A \,|\, (K_1 \times K_2)(< \chi_{X_j} >_{j\in\eta})(t_1, t_2) = (p_1, p_2))\}.$$

In other words this can be stated as follows

$$\lambda_{(p_1,p_2),A}(X_j)_{j\in\eta} = \lambda_{p_1}(X_j)_{j\in\eta} \times \lambda_{p_2}(X_j)_{j\in\eta},$$

where λ_{p_1} and λ_{p_2} are the singleton predicate liftings associated with p_1 for K_1 and associated with p_2 for K_2.

The semantics for such singleton predicate lifting is:

$$s \models_\alpha [\lambda_{(p_1,p_2)}](\varphi_j)_{j\in\eta} \text{ iff } \pi_1\alpha(s) \in \lambda_{p_1}([\varphi_j])_{j\in\eta} \text{ and } \pi_2\alpha(s) \in \lambda_{p_2}([\varphi])_{j\in\eta}.$$

Remark 4.4.1. *Notice that the predicate liftings for the product are associated with sets $C \subseteq K_1 2 \times K_2$. If the set C is the product of two sets we can give a nice characterization as we did in the case of singleton liftings. If not a characterization using the predicate liftings of the factors is much more obscure. Singleton liftings solve this problem because now the predicate lifting associated with C can be represented as the union of the predicate liftings of its elements.*

4.4.7 $Hom(D, K))$ functors

Now we will compose a Kripke polynomial functor, say K, with a homomorphism functor, say $Hom(D, -)$. A coalgebra for this functor is a function $\alpha : A \longrightarrow Hom(D, KA)$.

A singleton predicate lifting will be associated with a function $P : D \longrightarrow K(2^\eta)$. The predicate lifting associated with $\{P\}$ acts as follows: Given a family of subsets $(X_j)_{j\in\eta}$ of A

$$\lambda_{P,A}(X_j)_{j\in\eta} = \{h : D \longrightarrow TA \,|\, K([X_j]_{j\in\eta})h = P\}.$$

The semantic for such predicate lifting is:

$$s \models_\alpha [\lambda_P](\varphi_j)_{j\in\eta} \text{ iff } (\forall d \in D)(\alpha(s)(d) \in \lambda_{P(d)}(X_j)_{j\in\eta}),$$

where $\lambda_{P(d)}$ is the singleton predicate lifting associated with $P(d)$ for K.

4.4.8 $\mathcal{P}K$ functors

A coalgebra for this functor can be seen as a Kripke frame of type K, i.e is a function $\alpha : A \longrightarrow \mathcal{P}KA$. A singleton lifting will be associated with a set $P \subseteq K(2)$. Equation 4.2 tell us that the function of $\lambda_{(P,A)} : 2^A \longrightarrow 2^{\mathcal{P}KA}$ acts as follows: A set $X \subseteq A$ is mapped to

$$\lambda_{(P,A)}(X) = \{U \subseteq KA \,|\, K(\chi_X)[U] = P\},$$

where $K(\chi_X)[U]$ is the direct image of U under the function $K(\chi_X)$. Notice that $K(\chi_X) : KA \longrightarrow K2$, therefore its direct image goes from $\mathcal{P}KA$ to $\mathcal{P}K2$. Using this we can see that the elements in $\lambda_{(P,A)}(X)$ are the sets $U \subseteq KA$ such that

$$(\forall u \in U)(\exists p \in P)(K(\chi_X)(u) = p) \wedge \forall(p \in P)(\exists u \in U)(K(\chi_X)(u) = p).$$

Now we will provide a characterization that will be used to translate the predicate liftings for a functor of the form $\mathcal{P}K$ into the Moss' language for the same functor.

We will use capital P to denote subsets of $K2$ and we will use small p to denote elements of $K2$. Consider the predicate lifting associated with $P = \{p\}$, where $p \in K2$. The A-component of this predicate lifting goes from 2^A to $2^{\mathcal{P}KA}$. Rewriting the previous characterization the elements of $\lambda_{\{p\}}(X)$ are the sets $U \subseteq KA$ such that

$$(\forall u \in U)(K(\chi_X)(u) = p) \wedge (\exists u \in U)(K(\chi_X)(u) = p).$$

Now consider the predicate lifting associated with $p \in K2$ for the functor K. We already know that the A-component $\lambda_p : 2^A \longrightarrow 2^{KA}$ acts as follows

$$\lambda_p(X) = \{u \in KA \,|\, K(\chi_X)(u) = p\}.$$

Using this we can see that the formula

$$(\forall u \in U)(K(\chi_X)(u) = p)$$

is equivalent to the formula

$$U \subseteq \lambda_p(X).$$

The formula

$$(\exists u \in U)(K(\chi_X)(u) = p)$$

is expressing the fact that U is not empty. We conclude that the singleton lifting $\lambda_{\{p\}} : 2^A \longrightarrow 2^{\mathcal{P}KA}$ can be characterized in terms of λ_p as follows:

$$\lambda_{(\{p\},A)}(X) = \{U \subseteq TA \,|\, U \neq \emptyset \wedge U \subseteq \lambda_p(X)\}.$$

More general, we can see that for $P \subseteq K2$ the formula

$$(\forall u \in U)(\exists p \in P)(K(\chi_X)(u) = p)$$

is equivalent to the formula

$$U \subseteq \bigcup_{p \in P} \lambda_p(X).$$

We can also see that the formula

$$\forall(p \in P)(\exists u \in U)(K(\chi_X)(u) = p)$$

is equivalent to the formula

$$(\forall p \in P)(U \cap \lambda_p(X) \neq \emptyset).$$

Using that we can characterize $\lambda_{(P,A)} : 2^A \longrightarrow 2^{\mathcal{P}KA}$ in terms of the predicate liftings for K associated with the elements of P as follows

$$\lambda_{(P,A)}(X) = \{U \subseteq KA \,|\, U \subseteq \bigcup_{p\in P} \lambda_p(X) \,\wedge\, (\forall p \in P)(U \cap \lambda_p(X) \neq \emptyset)\},$$

where a $\lambda_p : 2^A \longrightarrow 2^{KA}$ is the singleton lifting associated with p for the functor K.

4.5 Coalgebraic Modal Languages Algebraically

On Chapter 3 we showed that Moss' language can also be represented as the initial algebra of the functor $\mathcal{P}_\kappa + T_\kappa$. In this section we will show that a similar characterization can be given for coalgebraic modal languages, i.e they can be understood in a more categorical context. First we show how to see predicate liftings as operators over sets.

As we said before there is another intuition, to understand predicate liftings. Namely predicate liftings convert coalgebras into neighborhood frames. The components of a predicate lifting λ,

$$\lambda_A : 2^A \longrightarrow 2^{TA},$$

are uniquely determined by its exponential adjoints,

$$\widetilde{\lambda_A} : TA \longrightarrow 2^{2^A}.$$

This functions are defined as follows

$$\widetilde{\lambda_A}(t) = \{X \subseteq A \,|\, t \in \lambda_A(X)\}.$$

Given a T-coalgebra α we can define a neighborhood model by composition

$$\widetilde{\lambda_A}\alpha : A \longrightarrow 2^{2^A}.$$

Then predicate liftings can be understand as a way to transform an obscure, and not really known, T-coalgebra into a more familiar structure like neighborhood models are.

Now notice that the exponential adjoint associated with the neighborhood model of a predicate lifting is a function

$$f_\alpha : 2^A \longrightarrow 2^A$$

defined as follows,

$$f_\alpha(X) = \{s \in A \,|\, \alpha(s) \in \lambda_A(X)\} = \alpha^{-1}\lambda_A(X).$$

This shows how to present predicate liftings as algebraic operators.

Generally speaking, given a T-coalgebra (A, α) and a predicate lifting $\lambda : 2^{(-)\times\eta} \longrightarrow 2^{T(-)}$, we define $\lambda_\alpha : \mathcal{P}A \longrightarrow \mathcal{P}A$ to be the following composite

$$\begin{tikzcd} 2^{A\times\eta} \arrow[rr, "\lambda_A"] \arrow[dr, "\lambda_\alpha"'] & & 2^{TA} \arrow[dl, "\alpha^{-1}"] \\ & 2^A & \end{tikzcd} .$$

Furthermore it can be shown that in fact we have

Lemma 4.5.1. *The functions λ_α, as defined above, have the following property: In the following schema*

$Coalg(T)$	$Alg(I)$
$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\alpha"'] & B \arrow[d, "\beta"] \\ TA \arrow[r, "T_f"'] & TB \end{tikzcd}$$	$$\begin{tikzcd} \check{\mathcal{P}}A \arrow[d, "\lambda_\alpha"'] & \check{\mathcal{P}}B \arrow[l, "f^{-1}"'] \arrow[d, "\lambda_\beta"] \\ \check{\mathcal{P}}A & \check{\mathcal{P}}B \arrow[l, "f^{-1}"] \end{tikzcd}$$

the diagram on the right commutes if the diagram on the left commutes.

4.5.1 Schröder functors

Following the idea used to define a functor M associated with Moss' language we now define a functor associated with a predicate lifting λ, and in general with a set of predicate liftings Λ.

Definition 4.5.2 (Schröder Functor)**.** *Given a predicate lifting $\lambda : 2^{(-)\times\eta} \longrightarrow 2^{T(-)}$, we define a functor*

$$S(\lambda) : Coalg(T)^{op} \longrightarrow Alg(\mathcal{P} + \mathcal{I}^\alpha + \mathcal{I})$$

in the following way. On objects: Given a T-coalgebra (A, α), the functor $S(\lambda)$ maps this coalgebra to

$$[\delta_A, \lambda_\alpha, \neg_A] : \mathcal{P}\mathcal{P}A + \mathcal{P}A + \mathcal{P}A \longrightarrow \mathcal{P}A,$$

where δ_A and $\neg_A$ are the functions defined on Chapter 3 representing conjunctions, see page 23, and negations, see page 32. On arrows: The functor $S(\lambda)$ maps a morphism $f : (A, \alpha) \longrightarrow (B, \beta)$ to

$$f^{-1} : (\mathcal{P}B, [\delta_B, \lambda_\beta, \neg_B]) \longrightarrow (\mathcal{P}A, [\delta_A, \lambda_\alpha, \neg_A]).$$

Given a set of predicate liftings Λ*, we define a functor*

$$S(\Lambda) : Coalg(T)^{op} \longrightarrow Alg(\mathcal{P} + \Lambda + \mathcal{I})$$

in a similar way. Here $\mathcal{P} + \Lambda + I$ *states for* $\mathcal{P} + \coprod_{\lambda \in \Lambda} I^{\eta_\lambda} + I$*, where* η_λ *is the arity of* λ*.*

Remark 4.5.3. *Notice that one of the reasons why the definition above makes sense, i.e Schröder functors are in fact functors, is that predicate liftings are natural transformation.*

Now we can redefine coalgebraic modal languages.

Definition 4.5.4 (Coalgebraic Modal languages II)**.** *Given a* κ*-accessible functor* $T : Set \longrightarrow Set$*, we define the* κ-coalgebraic modal language*, written* $\mathcal{L}^\kappa_T(\Lambda)$*, to be the carrier of* $(\mathcal{L}^\kappa_T(\Lambda), \bigwedge, \Lambda, \neg)$*, where this later structure is an initial algebra for the functor* $\mathcal{P}_\kappa + \Lambda + I$*, defined above.*

Using this and Schröder functors we redefine the satisfaction relation for coalgebraic modal language as follows: Let $T : Set \longrightarrow Set$ *be a functor that preserves weak pullbacks and let* (A, α) *be a* T_κ*-coalgebra. We define*

$$[-]_\alpha : \mathcal{L}^\kappa_T(\Lambda) \longrightarrow \mathcal{P}A$$

to be the unique arrow $[-]_\alpha : (\mathcal{L}^\kappa_T(\Lambda), [\bigwedge, \Lambda, \neg]) \longrightarrow (\mathcal{P}A, [\wedge_\alpha, \Lambda, \neg_\alpha])$ *that exists in* $Alg(\mathcal{P}_\kappa + \Lambda + I)$*. We define* $\models_\alpha$ *as follows:*

$$s \models_\alpha \varphi \textit{ iff } s \in [\varphi]_\alpha.$$

Using Schröder functors the following is immediate

Theorem 4.5.5. *Coalgebraic languages language associated with a functor* T_κ *are adequate with respect to behavioral equivalence.*

Unfortunately not all coalgebraic modal languages with only unary predicate liftings have the Hennessy-Milner property with respect to behavioral equivalence, for example modal logic for $\mathcal{P}$-coalgebras. This problem is solved assuming κ-accessibility and imposing some properties over the parameter Λ.

In [10] Schröder defines separating sets of predicate liftings as follows.

Definition 4.5.6 (Separating sets)**.** *A set* Λ *of predicate liftings for* T *is* separating *iff for each set* A*, the associated source of transposites*

$$\left(\widetilde{\lambda_A} : TA \longrightarrow 2^{((2^A)^\eta)}\right)_{\lambda \in \Lambda}$$

is jointly injective for each set A*. That is every* $t \in TA$ *is uniquely determined by the set*

$$\{(\lambda, X) \in \Lambda \times (2^A)^\eta \mid t \in \lambda_A(X)\}.$$

We say that set $C \subseteq T2$ *is separating if the associated set of predicate liftings is separating.*

In other word a set of predicate liftings is separating if it can differentiate elements in TA, i.e if $t \neq t'$ then there exists $\lambda \in \Lambda$ and $X \subseteq A$ such that

$$t \in \lambda_A(X) \text{ and } t' \notin \lambda_A(X).$$

Using separating sets of predicate liftings Schröder in [10] showed that under the condition of separability, coalgebraic modal languages are expressive.

Theorem 4.5.7. *Let T be a κ-accessible functor and let Λ be a separating set of predicate liftings, possibly polyadic, for T. Then $\mathcal{L}^{\kappa}(\Lambda)$ has the Hennessy-Milner property with respect to behavioral equivalence in the category of T-coalgebras.*

Now it happens to be the case that all accessible functors admit a separating set of predicate liftings, see [10].

Proposition 4.5.8. *If T is κ-accessible and preserves monos, then T admits a κ-bounded set of polyadic predicate liftings.*

As corollary we have

Corollary 4.5.9. *Let T be a κ-accessible functor and let Λ be the set of κ-bounded predicate liftings for T. Then $\mathcal{L}^{\kappa}(\Lambda)$ has the Hennessy-Milner property with respect to behavioral equivalence in the category of T-coalgebras.*

Chapter 5

Hennessy-Milner Logics and Final Coalgebras

In this chapter we will prove the following:

The existence of an expressive local language for behavioral equivalence is equivalent to the existence of a final coalgebra.

In the last section of this chapter we will use this construction to produce translations between Moss' language and coalgebraic modal languages.

Final Coalgebras are terminal objects in the category of coalgebras. Final coalgebras are particulary important because they code all the possible behaviors of states in the category $Coalg(T)$ and have the following coinductive property: Let (Z, ζ) be a final coalgebra and $s, s' \in Z$ then

$$s \underline{\leftrightarrow} s' \text{ iff } s = s' \text{ iff } s \sim s',$$

where the first relation is bisimilarity and the last one is behavioral equivalence.

If there is an expressive language $\mathcal{L}$ for bisimilarity, or behavioral equivalence, the previous equivalence can be extended to

$$s \equiv s' \text{ iff } s \underline{\leftrightarrow} s' \text{ iff } s = s' \text{ iff } s \sim s',$$

where the first relation is the logically equivalent relation. Notice that both adequacy and HM are needed to obtain the first equivalence.

In other words: if there is a final coalgebra (Z, ζ) and an adequate language $\mathcal{L}$ with the HM for behavioral equivalence then the states of the final coalgebra represent the truth classes of $\mathcal{L}$, see page 13.

In this chapter we will show that this is the only possibility, i.e. a language $\mathcal{L}$ is expressive iff we can define a final coalgebraic structure over the set of truth classes, written $(\mathcal{L}_{\models}, \zeta)$.

5.1 Simple Coalgebras

In this section we will mention concepts of Universal Algebra. We do not enter in details because they are not needed for the formal development of this thesis, but tho be familiar with those concepts might help the intuition. We encourage the interested reader to consult a book in Universal algebra. We recommend [3].

Congruences are one of the central objects in Universal Algebra. An equivalence relation θ on a set A, where A is the underlying set of some algebra, is called a congruence if the algebraic structure can be transferred to A/θ to make the quotient map, e_θ, into a homomorphism. Now any function has a *kernel* equivalence relation on its domain, namely

$$Ker f = \{(x, y) \mid f(x) = f(y)\}.$$

Furthermore, when f is a homomorphism of algebras this is a congruence. In fact in universal algebra the congruences are just the kernels of homomorphisms. An algebra is *simple* if it has no non-trivial congruences relations.

Suppose now that A is the state set of a T-coalgebra, say α. What does it take to make the quotient set A/θ into a coalgebra? The answer, see [5] or [6], is:

Theorem 5.1.1. *Let (A, α) be a T-coalgebra, let θ be an equivalence relation over A and $e_\theta : A \longrightarrow A/\theta$ be the canonical map. The following are equivalent:*

- *There is a unique structural map $d_\theta : A/\theta \longrightarrow T(A/\theta)$ such that the following diagram*

$$\begin{tikzcd} A \arrow[r, "e_\theta"] \arrow[d, "\alpha"'] & A/\theta \arrow[d, "d_\theta"] \\ TA \arrow[r, "T(e_\theta)"'] & T(A/\theta) \end{tikzcd}$$

  commutes.

- *$\theta \subseteq Ker(T(e_\theta)\alpha)$.*

- *The relation θ is the kernel of some T-coalgebra morphism with domain (A, α).*

Based on the previous theorem we define the concept of congruence.

Definition 5.1.2. *An equivalence relation, say θ, over the state set of a T-coalgebra, α, is called a* congruence *if*

$$\theta \subseteq Ker(T(e_\theta)\alpha),$$

where $e_\theta : A \longrightarrow A/\theta$ is the canonical map.

Not all bisimulations are congruences, that because some bisimulations are not equivalence relations. In general it is not the case that all congruences are bisimulations, unless the functor preserves weak kernels. There is a counterexample in [6]. Never the less, see [6], it can be shown that for any functor, any bisimulation can be extended to a smallest congruence containing it.

Proposition 5.1.3. *For every bisimulation (R, ρ) there exists the smallest congruence relation, $\langle R \rangle$, containing R.*

Proof. By definition the projections $\pi_1, \pi_2 : \rho \longrightarrow \alpha$ are T-morphism. Let $e : \alpha \longrightarrow \gamma$ be a coequalizer of π_1, π_2 in $Coalg(T)$. By Theorem 5.1.1 $Ker(e)$ is a congruence because e is a morphism in $Coalg(T)$. By definition of coequalizers it contains R and it is the smallest congruence containing R. □

In the category Set every coequalizer produces an equivalence relation and viceversa. This implies the following corollary.

Corollary 5.1.4. *Every bisimulation that is an equivalence relation is a congruence.*

It is shown in [6] that the set of all congruences is a complete lattice.

Theorem 5.1.5. *The set of all congruences on a coalgebra (A, α) is a complete lattice. The supremum is given by*

$$\bigvee \theta_j = (\bigcup \theta_j)^*,$$

where $(-)^*$ is the transitive closure. The infimum is given by

$$\bigwedge \theta_j = Con[\bigcap \theta_j],$$

where $Con[R]$ is the supremum of all congruences contained in R.

The smallest element of the lattice of congruences on (A, α) is trivial, i.e it is the identity, Δ_A. But unlike the universal algebra case the largest element Υ_A, may be, in general, a proper subset of $A \times A$. For example in the case of the constant functor a pair of states (s, s') in a coalgebra $\alpha : A \longrightarrow D$ belongs to some congruence iff $\alpha(s) = \alpha(s')$. However the largest congruence has the following uniqueness property:

Theorem 5.1.6. *If Υ_A is the largest congruence on α then for every coalgebra β there is at most one morphism $f : \beta \longrightarrow \alpha/\Upsilon_A$*

Proof. Assume $f_1, f_2 : \beta \longrightarrow \alpha/\Upsilon_A$ are two morphisms. Let $e : \alpha/\Upsilon_A \longrightarrow \xi$ coequalize them in $Colag(T)$. By Theorem 5.1.1 the relation $Ker(ef_\Upsilon)$, where f_Υ is the quotient map, is a congruence on α. Therefore $Ker(ef_\Upsilon) \subseteq \nabla_A$. In other words, for every $a, a' \in A$

$$ef_\nabla(a) = ef_\nabla(a') \text{ implies } a\nabla a'.$$

Fix $b \in B$ and let $a \in f_1(b), a' \in f_2(b)$. By definition $f_\nabla(a) = f_1(b)$ and $f_\nabla(a') = f_2(b)$. Then we have

$$ef_\nabla(a) = ef_1(b) = ef_2(b) = ef_\nabla(a').$$

Hence by the previous observation $a\nabla a'$, therefore $f_1(b) = f_2(b)$. This concludes the proof. □

The previous theorem shows that coalgebras of the form α/Υ_A are weakly terminal, i.e almost terminal but the existence condition is missing. This lead us to the concept of simple coalgebras.

Definition 5.1.7. *A coalgebra (A, α) is called* simple *if its largest, and hence only, congruence is Δ_A. In other words $\Delta_A = \Upsilon_A$.*

Simple coalgebras can also be characterized in the following way

Theorem 5.1.8. *For a coalgebra (A, α), the following are equivalent.*

1. *α is a simple coalgebra.*
2. *Every morphism with domain α is injective.*
3. *For every coalgebra β there is at most one morphism $f : \beta \longrightarrow \alpha$.*
4. *Every epimorphism with domain α is an isomorphism.*

Proof. The proof is almost direct from the definitions. The interested reader can also consult **??**. □

Based on this theorem we obtain the following relation between congruences, final coalgebras and simplicity.

Corollary 5.1.9.
- *For any coalgebra α, α/Υ_A is simple.*
- *Every final coalgebra is simple.*

Using this corollary we can also prove the following.

Theorem 5.1.10. *Assume a final coalgebra ζ exists. Then a coalgebra α is simple iff it is isomorphic to a subcoalgebra of ζ.*

5.2 From Final Coalgebras to Hennessy-Milner Logics

Proposition 5.2.1. *Suppose T has a final coalgebra (Z, ζ), with f_α being the unique morphism from α to ζ for each T-coalgebra (A, α). Then for any pointed coalgebras $(\alpha, s), (\beta, s')$,*

- *$s \underline{\leftrightarrow}_{\alpha\beta} s'$ implies $f_\alpha(s) = f_\beta(s')$.*
- *If T has transitive bisimilarity, then $f_\alpha(s) = f_\beta(s')$ implies $s \underline{\leftrightarrow}_{\alpha\beta} s'$.*

Proof. (1): If $s \underline{\leftrightarrow}_{\alpha\beta} s'$, then (s, s') belongs to some bisimulation R from α to β. Hence there exists a coalgebra structure, ρ, on R such that the following diagram

$$\begin{tikzcd}
\alpha \arrow[dr, "f_\alpha"'] & \rho \arrow[l, "\pi_A"'] \arrow[r, "\pi_B"] \arrow[d, "f_\rho"] & \beta \arrow[dl, "f_\beta"] \\
 & \zeta &
\end{tikzcd}$$

commutes. Form that we conclude

$$f_\alpha(s) = f_\alpha \pi_A(s, s') = f_\beta \pi_B(s, s') = f_\beta(s').$$

(2): Morphism are functional bisimulations, therefore

$$s \underline{\leftrightarrow} f_\alpha(s) = f_\beta(s') \underline{\leftrightarrow} s'.$$

Since T has transitive bisimilarity we conclude $s \underline{\leftrightarrow} s'$. □

Theorem 5.2.2. *If T has a final coalgebra then there exists an adequate local language $\mathcal{L}$ with the Hennessy-Milner property with respect to behavioral equivalence.*

Proof. Assume T has a final coalgebra (Z, ζ), with f_α being the unique morphism from α to ζ for each T-coalgebra (A, α).

We define a local language by taking $\mathcal{L} = Z$, so that $\mathcal{L}$ is small, and putting

$$s \models_\alpha \varphi \text{ iff } f_\alpha(s) = \varphi.$$

By definition $\mathcal{L}$ is adequate. On the other hand we defined $\mathcal{L}(\alpha, s) = \{f_\zeta(s)\}$, hence $\mathcal{L}(\alpha, s) = \mathcal{L}(\beta, s')$ iff $f_\alpha(s) = f_\beta(s')$. Hence logically equivalent states are behavioral equivalent, this concludes the proof. □

Corollary 5.2.3. *If T has transitive bisimilarity and a final coalgebra then there exists a local language $\mathcal{L}$ with the Hennessy-Milner property with respect to bisimilarity.*

Proof. Following the construction of the previous theorem we have

$$s \underline{\leftrightarrow} f_\alpha(s) = f_\beta(s') \underline{\leftrightarrow} s'.$$

If T has transitive bisimilarity we conclude $s \underline{\leftrightarrow} s'$. □

5.3 From Hennessy-Milner Logics to Final Coalgebras

In this section we present an elementary construction of final coalgebras by Clemens Kupke and the author directly over the set $\mathcal{L}_\models$, see page 13, of the truth- classes.

Theorem 5.3.1. *If there exists an adequate local language, say $\mathcal{L}$, for T that has the Hennessy-Milner property with respect to behavioral equivalence then T has a final coalgebra.*

Construction. We define a structural map, $\zeta : \mathcal{L}_\models \longrightarrow T\mathcal{L}_\models$, directly over the set of truth classes. Notice that the satisfaction relation is a function

$$\models_\alpha : A \longrightarrow \mathcal{L}_\models.$$

By definition for any $\Phi \in \mathcal{L}_\models$ there exists a pointed coalgebra (α, s) such that $\models_\alpha (s) = \Phi$. Using this we define

$$\zeta(\varphi) := T(\models_\alpha)\alpha(s).$$

□

First we have to show that the function ζ is well defined, i.e the value of ζ does not depend on the choice of α neither in the choice of s. First we show it is well defined under morphisms proving the following lemma.

Lemma 5.3.2. *Let $f : \alpha \longrightarrow \beta$ be a coalgebraic morphism, then for any $s \in A$*

$$T(\models_\alpha)\alpha(s) = T(\models_\beta)\beta f(s).$$

Proof. By the adequacy of $\mathcal{L}$ the following diagram

$$\begin{tikzcd} A \arrow[d, "f"'] \arrow[dr, "\models_\alpha"] & \\ B \arrow[r, "\models_\beta"'] & \mathcal{L}_{\models} \end{tikzcd}$$

commutes. Then, since T is a functor,

$$T(\models_\beta)T(f) = T(\models_\alpha).$$

Using this and the fact that f is a morphism we obtain:

$$T(\models_\beta)\beta f(s) = T(\models_\beta)T_f\alpha(s) = T(\models_\alpha)\alpha(s)$$

This concludes the proof. $\square$

Now we can prove

Claim 5.3.3. *The function $\zeta : \mathcal{L}_{\models} \longrightarrow T\mathcal{L}_{\models}$, defined above, is well defined. Explicitly it is: For any coalgebras α and β coalgebras and states $s \in A, s' \in B$ such that $\models_\alpha (s) = \models_\beta (s')$ the following holds*

$$T(\models_\alpha)\alpha(s) = T(\models_\beta)\beta(s').$$

Proof. Let α, β be coalgebras and $s \in A, s' \in B$ be logically equivalent. Therefore, by the HM, we conclude $s \sim s'$. By definition of behavioral equivalence it implies that there is a coalgebra (G, γ) and morphisms $f : \alpha \longrightarrow \gamma$ and $g : \beta \longrightarrow \gamma$ such that $f(s) = g(s')$. By the previous lemma we conclude

$$T(\models_\alpha)\alpha(s) = T(\models_\gamma)\gamma f(s) = T(\models_\gamma)\gamma g(s) = T(\models_\beta)\beta(s').$$

This concludes the proof. $\square$

We remark that our definition of ζ is based in the following diagram

$$\begin{tikzcd} A \arrow[r, "\models_\alpha"] \arrow[d, "\alpha"'] & \mathcal{L}_{\models} \arrow[d, "\zeta"] \\ TA \arrow[r, "T(\models_\alpha)"'] & T(\mathcal{L}_{\models}), \end{tikzcd}$$

i.e we want this diagram to commute. Since we did not know ζ we defined it is such a way that the diagram commutes, we followed the lower path. As we said before we had to show that ζ was well defined. Once we have it we conclude.

Corollary 5.3.4. *For any coalgebra α, the satisfaction function $\models_\alpha : \alpha \longrightarrow \zeta$ is a morphism.*

We now show that the coalgebra $(\mathcal{L}_{\models}, \zeta)$ is simple.

Claim 5.3.5. *The coalgebra $(\mathcal{L}_{\models}, \zeta)$ is simple.*

Proof. Let θ be a congruence on $\mathcal{L}_{\models}$, with e_θ its canonical map. We want to show that it is the identity. By definition of congruence we know

$$\theta \subseteq Ker(T(e_\theta)\zeta),$$

i.e. for every $\varphi, \psi \in \mathcal{L}_{\models}$

$$\varphi\theta\psi \text{ implies } T(e_\theta)\zeta(\varphi) = T(e_\theta)\zeta(\psi).$$

Assume φ is the truth class of a state s in a coalgebra α and that ψ is the truth class of a sate s' in a coalgebra β and assume $\varphi\theta\psi$. If we manage to show $s \sim s'$ by the adequacy of $\mathcal{L}$ we are done. Our assumption implies,

$$\models_\alpha (s) = \varphi \text{ and } \models_\beta (s') = \psi.$$

Therefore using the assumption $\varphi\theta\psi$ we obtain

$$e_\theta \models_\alpha (s) = e_\theta(\varphi) = e_\theta(\psi) = e_\theta \models_\beta (s').$$

since $\models_\alpha$ and $\models_\beta$ are morphism, by the previous corollary, we conclude $s \sim s'$ and this concludes the proof of the claim. □

By the previous corollary and Theorem 5.1.8 we conclude $(\mathcal{L}_{\models}, \zeta)$ is a final coalgebra. This concludes the proof of Theorem 5.3.1.

We have just showed that if an adequate language $\mathcal{L}$ has the HM with respect to behavioral equivalence, then we can define a final coalgebraic structure $(\mathcal{L}_{\models}, \zeta)$ such that for any coalgebra α the function $\models_\alpha$: $\alpha \longrightarrow \zeta$ is the only morphism. Now we show that it is in fact the only possibility.

Theorem 5.3.6. *Let $\mathcal{L}$ be a local language for coalgebras. If there is a final coalgebraic structure $(\mathcal{L}_{\models}, \zeta)$ such that for any coalgebra, α, the function $\models_\alpha$: $\alpha \longrightarrow \zeta$ is the only morphism, then $\mathcal{L}$ is adequate and has the Hennessy Milner property with respect to behavioral equivalence.*

Proof. Adequacy: First notice that under the existence of a final coalgebra, two states s, s', in coalgebras α and β, respectively, are behavioral equivalent iff they are mapped to the same state in the final coalgebra. Assume $s \sim s'$, then by hypotheses we conclude $\models_\alpha (s) = \models_\beta (s')$. That is $\mathcal{L}$ is adequate.

HM: Notice that two states s, s', in coalgebras α and β respectively, are logically equivalent iff $\models_\alpha (s) = \models_\beta (s')$. Since each $\models_\alpha$ is a morphism, we conclude $\mathcal{L}$ has the Hennessy-Milner property with respect to behavioral equivalence. □

5.4 Non-compositional translations

Now we will use the construction of Theorem 5.3.1, on the previous section, to produce translations between Moss' language and Coalgebraic modal languages.

The idea is the following: On chapter 3 we proved that Moss' language, written $\mathcal{L}_T^\kappa$, for a κ-accessible functor is a set and has the Hennessy-Milner property with respect to behavioral equivalence. On chapter 4 we proved that every κ-accessible functor that preserves monos admits a set of separating predicate liftings, say Λ, hence the language $\mathcal{L}_T^\kappa(\Lambda)$ has the Hennessy-Milner property with respect to behavioral equivalence. Using theorem 5.3.1, we will construct final coalgebras ζ_M and ζ_Λ, respectively. Then using Moss functor over ζ_Λ we obtain a translation $l : \mathcal{L}_T^\kappa \longrightarrow \mathcal{L}_T(\Lambda)$. Using the Schröder functor over ζ_M we obtain a translation $m : \mathcal{L}_T^\kappa(\Lambda) \longrightarrow \mathcal{L}_T(\neg)$.

Theorem 5.4.1. *Let T be a κ-accessible functor that preserves weak pull backs, then there exists a translation*

$$l : \mathcal{L}_T^\kappa \longrightarrow \mathcal{L}_T(\Lambda),$$

where Λ is a separating set of predicate liftings. Notice that we are allowing arbitrary conjunctions on the language with predicate liftings.

Proof. From Chapter 4 we know that the language $\mathcal{L}_T^\kappa(\Lambda)$ is expressive. Using Theorem 5.3.1 we construct a final coalgebra (Z, ζ) associated with $\mathcal{L}_T^\kappa(\Lambda)$. We recall that the states of this final coalgebra are the truth classes of $\mathcal{L}_T^\kappa(\Lambda)$.

Using Moss' functor we map (Z, ζ) into the category $Alg(\mathcal{P}_\kappa + T)$. By the definition of Moss' satisfaction, the following diagram

$$\begin{tikzcd}
\mathcal{L}_T \arrow[r, "{[-]}"] & \mathcal{P}Z \\
\mathcal{P}\mathcal{L}_T + T\mathcal{L}_T \arrow[u] \arrow[r] & \mathcal{P}\mathcal{P}Z + T\mathcal{P}Z \arrow[u]
\end{tikzcd}$$

commutes, where the vertical right arrow is the complex algebra associated with (Z, ζ).

We define a function, $l : \mathcal{L}_T^\kappa \longrightarrow \mathcal{L}_T^\kappa(\Lambda)$, as follows:

$$l(\bigwedge \varphi) = \bigwedge l(\varphi),$$
$$l(\nabla_T \varphi) = \bigvee_{\widetilde{s'} \in [\nabla_T \varphi]} \bigwedge \mathcal{L}_T^\kappa(\Lambda)(\zeta, \widetilde{s'}).$$

We now claim that the function l, defined above, is a translation, i.e for every coalgebra (α, s) and any formula $\varphi \in \mathcal{L}_T$ the following holds

$$s \Vdash_\alpha \varphi \text{ iff } s \models_\alpha l(\varphi).$$

We will only take care about the operator ∇_T.

From left to right: Let $f_\alpha : \alpha \longrightarrow \zeta$ be the unique terminal morphism in $Colag(T)$. Since Moss' language is adequate we conclude $f_\alpha(s) \Vdash_\zeta \nabla_T \varphi$. On the other hand since $\mathcal{L}_T^\kappa(\Lambda)$ is also adequate we conclude

$$\mathcal{L}_T^\kappa(\Lambda)(\alpha, s) = \mathcal{L}_T^\kappa(\Lambda)(\zeta, f_\alpha(s)).$$

This gives the implication from left to right.

From right to left: Assume $s \models_\alpha \bigwedge \mathcal{L}_T^\kappa(\Lambda)(\zeta, \widetilde{s'})$ for some s' such that $\widetilde{s'} \Vdash_\zeta \nabla_T \varphi$. By the adequacy of Moss' language it is enough to show $\widetilde{s'} \sim s$. In order to show that, we will use the Hennessy-Milner Property of $\mathcal{L}_T^\kappa(\Lambda)$.

By definition truth classes we have

$$\psi \in \mathcal{L}_T^\kappa(\Lambda)(\alpha, s) \text{ iff } \psi \in \mathcal{L}_T^\kappa(\Lambda)(\zeta, \widetilde{s'}),$$

here we use the fact that $\mathcal{L}_T^\kappa(\Lambda)$ has negations. Therefore, using the Hennessy-Milner Property of $\mathcal{L}_T^\kappa(\Lambda)$, we conclude that s is behavioral equivalent to $\widetilde{s'}$. Since Moss' language is adequate we conclude

$$s \Vdash_\alpha \nabla_T \varphi.$$

This concludes the proof. □

From that we conclude

Corollary 5.4.2. *The Moss' modality, ∇_T, is expressible in $\mathcal{L}_T(\Lambda)$.*

Furthermore, based on chapter 4 we conclude

Corollary 5.4.3. *Let T be a κ-accessible functor that preserves monos, then there exists a set of predicate liftings, say Λ that can express the Moss' modality ∇.*

And clearly we have

Theorem 5.4.4. *For every regular cardinal κ The Moss language $\mathcal{L}_T$ is more expressive than $\mathcal{L}_T^\kappa(\Lambda)$.*

Now we can use the same procedure to translate predicate liftings

Theorem 5.4.5. *Let T be a κ-accessible functor that preserves weak pull backs, then there exists a translation*

$$m : \mathcal{L}_T^\kappa(\Lambda) \longrightarrow \mathcal{L}_T(\neg).$$

Notice that we are allowing arbitrary conjunctions on the Moss' language.

Proof. The construction and proof are the same as in Theorem 5.4.1, i.e the construction goes as follows:

By Theorem 3.2.16 the language $\mathcal{L}_T^\kappa$ is expressive. Using Theorem 5.3.1 we construct a final coalgebra (Z, ζ) associated with $\mathcal{L}_T^\kappa$. We recall that the states of this final coalgebra are the truth classes of $\mathcal{L}_T^\kappa$. Using Schröder's functor we map (Z, ζ) into the category $Alg(\mathcal{P}_\kappa + \Lambda)$.

We define a translation, $m : \mathcal{L}_T^\kappa(\Lambda) \longrightarrow \mathcal{L}_T(\neg)$, as follows:

$$\begin{aligned} m(\bigwedge \varphi) &= \bigwedge m(\varphi), \\ m(\neg\varphi) &= \neg m(\varphi), \\ m([\lambda]\varphi) &= \bigvee_{\widetilde{s'} \in [\lambda\varphi]} \bigwedge \mathcal{L}(\zeta, \widetilde{s'}), \text{ where } \lambda \in \Lambda. \end{aligned}$$

Using the procedure invented for the proof of Theorem 5.4.1 we conclude that m is in fact a translation. □

From that we conclude

Corollary 5.4.6. *Let T be a κ-accessible functor that preserves weak pullbacks. Every predicate lifting for T is expressible in $\mathcal{L}_T(\neg)$.*

finally we obtain

Theorem 5.4.7. *Let T be a κ-accessible functor that preserves weak pullbacks and monos, then the Moss language $\mathcal{L}_T(\neg)$ and the language $\mathcal{L}_T(\Lambda)$, where Λ is the set of all predicate liftings, are equally expressive via translations.*

Unfortunately the previous translations are not compositional and highly non constructive. Such translations are also not finitary. For example in the case of the power set functor the nabla operator would be translated as a formula with ω conjunctions. But we know that in this case the nabla operator has a nice finitary translation, see equation 3.3.4 on page 28 in Chapter 3. We will try to obtain more refined translations as a result. We particulary looking for translations that are finitary. We do not approach this problem at a general level but only in the case of Kripke polynomial functors, more specifically finitary Kripke polynomial functors.

Chapter 6

From Moss' Language to Predicate Liftings and Back

In this chapter, we will compare Moss' language with negations, written $\mathcal{L}^{\kappa}_{T}(\neg)$, and languages with predicate liftings, written $\mathcal{L}^{\kappa}_{T}(\Lambda)$, for a given Kripke polynomial functor T. We will define, when possible, translations between the two languages. Recall that a translation between two languages is a function $Trs : \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ that commutes with satisfaction.

The collection of *Kripke polynomial functors* is inductively defined as follows:

$$K := \mathcal{I} \,|\, D \,|\, K_0 + K_1 \,|\, K_0 \times K_1 \,|\, Hom(D, K) \,|\, \mathcal{P}K.$$

Here $\mathcal{I}$ is the identity functor, D is a constant functor with value D, $\mathcal{P}$ is the covariant power set functor, $Hom(D, -)$ is the homomorphism functor, products and coproducts are taken pointwise, see Chapter 2. Replacing $\mathcal{P}$ with the finite power set functor $\mathcal{P}_{\omega}$, and demanding D to be finite, we obtain the collection of *finitary Kripke polynomial functors.*

We are particulary interested in finitary translations. A translation $m : \mathcal{L}^{\kappa}_{T}(\Lambda) \longrightarrow \mathcal{L}^{\kappa}_{T}(\neg)$ is said finitary if it maps finite formulas involving monadic predicate liftings to finite formulas of finitary Kripke polynomial functors. When possible, we will stress that our translations are in fact finitary.

We will define translations inductively over the complexity of the functor. In Chapter 3, we had to use Pattinson transformations to describe Moss' modality in the inductive cases. We have a similar problem when we want to carry out an inductive construction of translations. It is not enough to see translations only as functions. One more time we consider modalities, nablas or predicate liftings, not only act over formulas but rather over sets, i.e. they are natural transformations. Our plan is to translate this natural transformations instead of only formulas. We present a formal idea below. The intuitive idea is: We will obtain translations composing natural transformations. Notice that nabla operators, i.e the components of Pattinson transformations, are functions taking an argument of type $T\check{\mathcal{P}}A$ and producing a set. Divergent to that, predicate liftings transform subsets into subsets. If we want to translate such functions into each other we should translate two things: one, the assignation rule of the function itself, and two, the argument of the function. As we said, these two kinds of translations are formalized using composition of natural transformations. When our translation is obtained using composition of natural transformations, we call it a *natural translation.* Natural translations allow us to pass over the inductive steps because natural transformations are defined for all sets. The main problem is obtaining syntactic translations back from such natural transformations. A natural transformation that produces a syntactic translation is said to be *logically representable.* Using this technique we will prove:

Theorem 6.0.8. *Let Λ be the set of monadic predicate liftings for a Kripke polynomial functor T.*

- *Every formula of the language $\mathcal{L}^{\kappa}_{T}(\Lambda)$ can be expressed using formulas of $\mathcal{L}^{\kappa}_{T}(\neg)$.*

In particular we will show:

Corollary 6.0.9. *Let Λ be the set of monadic predicate liftings for a finitary Kripke Polynomial functor T.*

- *Every finite formula of the language $\mathcal{L}^{\kappa}_{T}(\Lambda)$ can be expressed using finite formulas of $\mathcal{L}^{\kappa}_{T}(\neg)$.*

In the following section we, detail the general procedure to translate predicate liftings. In order to provide a more concrete explanation, we will exemplify our procedure using the identity functor and the covariant power set functor before explaining the general theory. To give an idea of how to extend the translations over the inductive steps, we will use combinations of the power set functor and the identity functor.

A final comment before proceeding. In the following pages, we always assume the boolean operators are translated as usual, hence we will only present our translations in the case of the modal operators.

6.1 Translating Predicate Liftings

At a first glance, translating all predicate liftings might seem an enormous amount work. We will use singleton liftings to simplify our work. Theorem 4.3.6, see page 44, implies that it is enough to translate singleton liftings. Notice that every finitary predicate lifting for finitary Kripke polynomial functors can be expressed using only finitely many singleton liftings. This procedure will allow us to restrict our translations to finitary translations. We should remark that this idea will not work if the set $T(2)$ is infinite. However, we will show by induction on the complexity of the functor that with each unary singleton lifting $\lambda_p\varphi$ there is a map $m_p : \mathcal{L}^{\kappa}_{T}(\neg) \longrightarrow T\mathcal{L}^{\kappa}_{T}(\neg)$ satisfying some nice properties that help us to produce syntactic translations, i.e. in the case of the singleton liftings it is enough to translate arguments. As stated, we will do it composing natural transformations.

Before explaining the general theory let's use examples. We start with the identity functor.

6.1.1 The identity functor

A coalgebra for the identity functor is a function $\alpha : A \longrightarrow A$. Moss' Language for this functor can be found on page 27. Coalgebraic modal logic for this functor can be found on page 45.

As we saw in section 4.4.1 of Chapter 4, the singleton liftings for the identity functor are associated to $\top$ and $\bot$. In that section we used $[1]$ for the first singleton lifting and $[\neg]$ for the other. In some diagram because of some LaTeX technicalities we will also use $\lambda_\top$ and $\lambda_\bot$ to denote these singleton liftings.

In the case of the identity functor the syntactic translation is immediate. We define a syntactical translation $m : \mathcal{L}_I(\Lambda) \longrightarrow \mathcal{L}_I(\neg)$ as follows:

$$\begin{aligned} m([1]\varphi) &= \nabla m(\varphi), \\ m([\neg]\varphi) &= \nabla \neg m(\varphi), \\ m([M]\varphi) &= \bigwedge \emptyset, \\ m([\emptyset]\varphi) &= \bigvee \emptyset. \end{aligned}$$

This translation is fine but is hiding its origin. We will show that this translation is obtained from a natural transformation. This translation will help us to explain our technique.

In order to obtain natural transformations from that translation, we notice that in this case the Moss' modality has the same semantics as the predicate lifting [1]. Let's play the blind translator for a bit and let's assume we can use the same formula in the Moss' language and in the coalgebraic language, keep in mind that this assumption is wrong, using that our translation can be written

$$m([1]\varphi) = \nabla\varphi.$$

We can read this new equality as follows: The predicate lifting [1] is the same as the operator ∇. If we see it in terms of satisfaction sets we can see that the following diagram

$$\begin{tikzcd} \breve{\mathcal{P}} \arrow[rr, "id"] \arrow[dr, "\lambda_{\top,\alpha}"'] & & \breve{\mathcal{P}} \arrow[dl, "\nabla_\alpha"] \\ & \breve{\mathcal{P}} & \end{tikzcd}$$

commutes for each coalgebra α. The natural transformation associated with [1] is the identity. Notice that the components of this natural transformation can be defined for every $\mathcal{P}_\kappa + I$-algebra.

Form this diagram we can obtain the syntactic translation back. Notice that the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[r, "id"] & \check{\mathcal{P}}A \\
\mathcal{L}_I(\neg) \arrow[u, "{[-]_\alpha}"] \arrow[r, "id"'] & \mathcal{L}_I(\neg) \arrow[u, "{[-]_\alpha}"']
\end{tikzcd}$$

commutes for every coalgebra α. Call the identity function on the bottom of the previous square $m_\top : \mathcal{L}_I(\neg) \longrightarrow \mathcal{L}_I(\neg)$. Using $m_\top$ we obtain the syntactic translation back as follows: Star defining a syntactic translation $m : \mathcal{L}_I(\Lambda) \longrightarrow \mathcal{L}_I(\neg)$ inductively. To translate formulas of the form $[1]\varphi$ use the following diagram

$$\begin{tikzcd}
\mathcal{L}_I(\Lambda) \arrow[r, "m"] \arrow[d, "\lambda_\top"'] & \mathcal{L}_I(\neg) \arrow[d, dashed] \arrow[dr, "m_\top"] & \\
\mathcal{L}_I(\Lambda) \arrow[r, "m"'] & \mathcal{L}_I(\neg) & \mathcal{L}_I(\neg) \arrow[l, "\nabla"]
\end{tikzcd}$$

Thus the translation of a formula $[1]\varphi$ is

$$m([1]\varphi) = \nabla m_\top m(\varphi) = \nabla m(\varphi),$$

and that is exactly what we had before. In other words in order to define a translation of a formula $[1]\varphi$ we assume that inductively we already know how to use the translation m over the formula φ and then we use Moss' modality over it.

This procedure also works for the other singleton lifting, in that case, notice that the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}} \arrow[rr, "\neg"] \arrow[dr, "\lambda_{\perp,\alpha}"'] & & \check{\mathcal{P}} \arrow[dl, "\nabla_\alpha"] \\
& \check{\mathcal{P}} &
\end{tikzcd},$$

commutes for every coalgebra α. The natural transformation associated with $[\neg]$ is the complement natural transformation. Again notice that the components of this natural transformation can be defined for every $\mathcal{P}_\kappa{+}I$-algebra. Furthermore notice that the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[r, "\neg_A"] & \check{\mathcal{P}}A \\
\mathcal{L}_I(\neg) \arrow[u, "{[-]_\alpha}"] \arrow[r, "\neg"'] & \mathcal{L}_I(\neg) \arrow[u, "{[-]_\alpha}"']
\end{tikzcd}$$

commutes for every coalgebra α, where the function at the bottom is the negation of the language, we use $m_\perp$ to represent that function. To obtain the translation back again we start defining a syntactic translation $m : \mathcal{L}_I(\Lambda) \longrightarrow \mathcal{L}_I(\neg)$

inductively. To translate formulas of the form $[\neg]\varphi$ we use the following diagram

$$\begin{tikzcd}
\mathcal{L}_I(\Lambda) \arrow[r, "m"] \arrow[d, "\lambda_\top"'] & \mathcal{L}_I(\neg) \arrow[d, dashed] \arrow[dr, "m_\perp"] & \\
\mathcal{L}_I(\Lambda) \arrow[r, "m"'] & \mathcal{L}_I(\neg) & \mathcal{L}_I(\neg) \arrow[l, "\nabla"]
\end{tikzcd}$$

Using this diagram we conclude that the translation of a formula $[\neg]\varphi$ is

$$m([1]\varphi) = \nabla m_\perp m(\varphi) = \nabla\neg m(\varphi).$$

Again we obtain the translation above. Before explaining the general theory we will show how the same procedure works for the covariant power set functor.

6.1.2 Power set functor

In this section, we will illustrate the general procedure explained in the next section. Our reader has, at least, three options: one, to continue reading following the linear order of the text. Two, to use a non-linear order skipping this section now, reading the next section first and then go back to read this section. Three, neither of the previous two. What we recommend is to read this section two times. At a first instance, read it now to develop an intuition of what is going to happen in the next section. But also do a parallel reading with the next section. In this section we treat two examples of *logical translators* in detail. The concept of logical translator can be found in the next section. The structure of each example is exactly the structure of next section. More explicitly for each example, we will define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$. Using τ we will show that we can express a predicate lifting

$$\lambda : \check{\mathcal{P}}\check{\mathcal{P}}\mathcal{P}$$

using Pattinson transformation

$$\nabla : \mathcal{P}\check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}\mathcal{P},$$

for the covariant power set functor, as follows

$$\lambda = \nabla\tau.$$

Moss Logic for the power set functor can be found on page 28. Coalgebraic modal logic for this functor can be found on page 47.

As stated, playing the blind translator, the universal and existential modality can be translated in terms of Moss' language as follows:

$$\Diamond\varphi = \nabla\{\varphi, \top\}; \quad \Box\varphi = \nabla\emptyset \vee \nabla\{\varphi\}.$$

In this section, we will show how the procedure used in the case of the identity functor can also be used to translate the singleton liftings of the covariant power set functor. The singleton liftings of the power set functor are associated with the sets

$$\emptyset,\ \{\top\},\ \{\perp\},\ \{\perp, \top\}.$$

The existential modality is not a singleton lifting, it is associated with the set $\{\{\top\},\{\bot,\top\}\}$, but the translation suggested above allows us to produce a translation using the same technique we will use for the singleton liftings. Hence to help the intuition we will explain our technique in the case of the existential modality and in the case of the predicate lifting associated with $\{\top\}$.

The existential modality

We start with the existential modality. The formula

$$\Diamond\varphi = \nabla\{\varphi,\top\}$$

shows us that in this case it is enough to translate the argument. Recall that the A-component of the existential modality maps a set $X \subseteq A$ to

$$\Diamond_A(X) = \{U \subseteq A \,|\, U \cap X \neq \emptyset\}.$$

In this section we will do three things:

1. We will define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$.
2. We will prove that for each set A the following holds

$$\Diamond_A = \nabla_A \tau_A.$$

   Recall that ∇_A is the notation for Pattinson transformations.
3. Extract the translation above from τ.

First we define a natural transformation

$$\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$$

as follows: the A-component of τ is a function

$$\tau_A : \check{\mathcal{P}}A \longrightarrow \mathcal{P}\check{\mathcal{P}}A.$$

This function maps a set $X \subseteq A$

$$\tau_A(X) = \{X, A\}.$$

As we said now we want to show that these functions are the components of a natural transformation. In order to show that we have to show that given a function $f : A \longrightarrow B$ and $X \subseteq B$ the following holds

$$\tau_A(f^{-1}(X)) = \mathcal{P}(f^{-1})\tau_B(X).$$

By definition we have

$$\tau_A(f^{-1}(X)) = \{f^{-1}(X), A\}.$$

On the other side of the equality we have

$$\mathcal{P}(f^{-1})\tau_B(X) = \mathcal{P}(f^{-1})(\{X, B\}).$$

By definition of $\mathcal{P}(f^{-1})$ we conclude that

$$\mathcal{P}(f^{-1})\tau_B(X) = \{f^{-1}(X), f^{-1}(B)\}.$$

Now notice that for every function $f : A \longrightarrow B$ the inverse image of B is A, using this we conclude the desired equality and then that τ is a natural transformation.

Now as stated we will prove $\Diamond = \nabla\tau$. In order to prove this we have to show that the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[rr, "\tau_A"] \arrow[dr, "\Diamond_A"'] & & \mathcal{P}\check{\mathcal{P}}A \arrow[dl, "\nabla_A"] \\
& \check{\mathcal{P}}A &
\end{tikzcd},$$

commutes for every set A. By definition of relation lifting for the covariant power set functor we know that $U \in \nabla_A\tau_A(X)$ iff $U \subseteq A$ and

$$(\forall u \in U)(u \in X \vee u \in A) \wedge (\exists u \in U)(u \in A) \wedge (\exists u \in U)(u \in X).$$

Notice that the first two conjunctions are always true if U is non empty, also notice that the last conjunction implies that U is non empty. Therefore we conclude

$$U \in \nabla_A\tau_A(X) \text{ iff } U \subseteq A \wedge (\exists u \in U)(u \in X) \text{ iff } U \in \Diamond_A(X).$$

We conclude $\Diamond = \nabla\tau$ as we wanted. Using the terminology of the next section we have just showed that τ is a natural translator for $\Diamond$.

Now we will extract the translation above using τ. Notice that given a $\mathcal{P}_\kappa + I$-algebra $\mathfrak{A} = (A, \wedge, \neg)$, we can define a function

$$\tau_{\mathfrak{A}} : A \longrightarrow \mathcal{P}A$$

imitating the same idea we used to define τ as follows: An element $x \in A$ is mapped to

$$\tau_{\mathfrak{A}}(x) = \{x, \top\},$$

where $\top = \wedge(\emptyset)$. Furthermore notice that this functions have the following property: Let $f : \mathfrak{A} \longrightarrow \mathfrak{B}$ be a morphism of $\mathcal{P}_\kappa + I$ algebras then the following diagram

$$\begin{tikzcd}
A \arrow[r, "\tau_{\mathfrak{A}}"] \arrow[d, "f"'] & \mathcal{P}A \arrow[d, "\mathcal{P}(f)"] \\
B \arrow[r, "\tau_{\mathfrak{B}}"'] & \mathcal{P}B
\end{tikzcd}$$

commutes. In order to prove this, unravelling all definitions, we have to show for $x \in A$

$$\{f(x), \wedge_{\mathfrak{A}}(\emptyset)\} = \mathcal{P}(f)(\{x, \wedge_{\mathfrak{B}}(\emptyset)\}).$$

By definition of $\mathcal{P}(f)$ we have

$$\mathcal{P}(f)(\{x, \wedge_{\mathfrak{B}}(\emptyset)\}) = \{f(x), f(\wedge_{\mathfrak{B}}(\emptyset))\}.$$

Since f is a morphism of $\mathcal{P}_\kappa + I$-algebras we have

$$f\wedge_{\mathfrak{B}} = \wedge_{\mathfrak{A}}\mathcal{P}(f).$$

Finally it is well known that

$$\mathcal{P}(f)(\emptyset) = \emptyset.$$

From that we conclude

$$\mathcal{P}(f)(\{x, \wedge_{\mathfrak{B}}(\emptyset)\}) = \{f(x), f(\wedge_{\mathfrak{B}}(\emptyset))\} = \{f(x), \wedge_{\mathfrak{B}}(\emptyset)\}.$$

This illustrates the commutativity of the square above. Using the terminology of next section we have defined a boolean transformation $\overline{\tau} : I \longrightarrow \mathcal{P}$.

Notice that if $\mathfrak{A} = (\check{\mathcal{P}}A, \wedge, \neg)$ is a $\mathcal{P}_\kappa + I$-algebra where $\wedge$ is the A-component of the natural transformation for conjunctions and $\neg$ is the A component of the natural transformation for negations, then

$$\tau_{\mathfrak{A}} = \tau_A.$$

In the terminology of the next section we say that $\overline{\tau}$ extends τ. We also say that τ is a logical translator.

Notice that $\mathcal{L}_{\mathcal{P}}(\neg)$ is a $\mathcal{P}_\kappa + I$ algebra. Hence we have a function

$$m_\Diamond = \tau_{\mathcal{L}_{\mathcal{P}}(\neg)} : \mathcal{L}_{\mathcal{P}}(\neg) \longrightarrow \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg)$$

such that

$$m_\Diamond(\varphi) = \{\varphi, \top\}.$$

We remark that the function $m_\Diamond$ has as domain the Moss' language and not the language of predicate liftings. Furthermore notice the the Moss' satisfaction $[-]_\alpha$ is a morphism of $\mathcal{P}_\kappa + I$-algebras hence the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[r, "\tau_A"] & \mathcal{P}\check{\mathcal{P}}A \\
\mathcal{L}_{\mathcal{P}}(\neg) \arrow[u, "{[-]_\alpha}"] \arrow[r, "m_\Diamond"'] & \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg) \arrow[u, "{\mathcal{P}([-]_\alpha)}"']
\end{tikzcd}$$

commutes for every coalgebra α.

Using all this we define a syntactic translation $m : \mathcal{L}_{\mathcal{P}}(\Lambda) \longrightarrow \mathcal{L}_{\mathcal{P}}(\neg)$ inductively. To translate formulas of the form $\Diamond\varphi$ we use the following diagram

$$\begin{tikzcd}
\mathcal{L}_{\mathcal{P}}(\Lambda) \arrow[r, "m"] \arrow[d, "\Diamond"'] & \mathcal{L}_{\mathcal{P}}(\neg) \arrow[d, dashed] \arrow[dr, "m_\Diamond"] & \\
\mathcal{L}_{\mathcal{P}}(\Lambda) \arrow[r, "m"'] & \mathcal{L}_{\mathcal{P}}(\neg) & \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg) \arrow[l, "\nabla"]
\end{tikzcd}$$

Using this diagram we can describe the translation of a formula $\Diamond\varphi$ as follows

$$m(\Diamond\varphi) = \nabla m_\Diamond m(\varphi) = \nabla\{m(\varphi), \top\}.$$

And it is exactly the translation that we wanted. Using the notation of the next section we have proved that the translation m is a natural translation.

Another example

We hope our reader is getting the idea. We will give one more detailed example before explaining the general procedure. We will study the predicate lifting associated with $\{\top\}$. We recall that the predicate lifting $\lambda_{\{\top\}}$ has as components

$$\lambda_A(X) = \{U \subset A \,|\, U \neq \emptyset \wedge U \subseteq X\}.$$

Using the basic modal language, a formula $\lambda_{\{\top\}}\varphi$ is equivalent to the formula $\Box\varphi \wedge \Diamond\varphi$. This time we will not give the translation from the beginning but we will produce it using the technique that is not explicit yet.

Again we will do three things.

1. We will define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$.
2. We will prove that for each set A the following holds

$$\lambda_{(\{\top\},A)} = \nabla_A \tau_A.$$

   Recall that ∇_A is the notation for Pattinson transformations.
3. We will extract a translation from τ.

First we define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$, associated with $\{\top\}$. The A-component of τ maps a set $X \subseteq A$ to

$$\tau_A(X) = \{X\}.$$

Now we will check that this defines a natural transformation. In order to show that we have to show that given a function $f : A \longrightarrow B$ and $X \subseteq B$ the following holds

$$\tau_A(f^{-1}(X)) = \mathcal{P}(f^{-1})\tau_B(X).$$

By definition we have

$$\tau_A(f^{-1}(X)) = \{f^{-1}(X)\}.$$

On the other side of the equality we have

$$\mathcal{P}(f^{-1})\tau_B(X) = \mathcal{P}(f^{-1})(\{X\}).$$

By definition of $\mathcal{P}(f^{-1})$ we conclude that

$$\mathcal{P}(f^{-1})\tau_B(X) = \{f^{-1}(X)\}.$$

We conclude the desired equality and then that τ is a natural transformation.

We can easily check that for each set A the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[rr, "\tau_A"] \arrow[dr, "\lambda_{(\{\top\},A)}"'] & & \mathcal{P}\check{\mathcal{P}}A \arrow[dl, "\nabla_A"] \\
& \check{\mathcal{P}}A &
\end{tikzcd}$$

commutes. Notice that in this case $U \in \nabla_A\tau_A(X)$ iff $U \subseteq A$ and

$$(\forall u \in U)(u \in X) \wedge (\exists u \in U)(u \in X).$$

Therefore it is immediate that

$$U \in \nabla_A \tau_A(X) \text{ iff } U \in \lambda_{(\{\top\},A)}(X).$$

In the terminology of the next section we have showed that τ is a natural translator for $\lambda_{\{\top\}}$.

Now for every $\mathcal{P}_\kappa + I$-algebra $\mathfrak{A} = (A, \wedge, \neg)$, we define a function

$$\tau_{\mathfrak{A}} : A \longrightarrow \mathcal{P}A$$

following the same idea. Explicitly an element $x \in A$ is mapped to $\tau_{\mathfrak{A}}(x) = \{x\}$. Furthermore these functions have the following property: Let $f : \mathfrak{A} \longrightarrow \mathfrak{B}$ be a morphism of $\mathcal{P}_\kappa + I$ algebras then the following diagram

$$\begin{tikzcd}
A \arrow[r, "\tau_{\mathfrak{A}}"] \arrow[d, "f"'] & \mathcal{P}A \arrow[d, "\mathcal{P}(f)"] \\
B \arrow[r, "\tau_{\mathfrak{B}}"'] & \mathcal{P}B
\end{tikzcd}$$

commutes. In order to prove that we have to show that for $x \in B$ the following holds

$$\{f(x)\} = \mathcal{P}(f)(\{x\}).$$

The follows immediately from the definition of $\mathcal{P}(f)$. Using the terminology of the next section we have defined a boolean transformation $\overline{\tau} : I \longrightarrow \mathcal{P}$.

Notice that if $\mathfrak{A} = (\check{\mathcal{P}}A, \wedge, \neg)$ is a $\mathcal{P}_\kappa + I$-algebra where $\wedge$ is the A-component of the natural transformation for conjunctions and $\neg$ is the A component of the natural transformation for negations, then

$$\tau_{\mathfrak{A}} = \tau_A.$$

In the terminology of the next section we say that $\overline{\tau}$ extends τ. We also say that τ is a logical translator.

Since $\mathcal{L}_{\mathcal{P}}(\neg)$ is a $\mathcal{P}_\kappa + I$ algebra, in particular we have a function $m_{\{\top\}}$: $\mathcal{L}_{\mathcal{P}}(\neg) \longrightarrow \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg)$, which maps a formula $\varphi \in \mathcal{L}_{\mathcal{P}}(\neg)$ to $\{\varphi\}$. Notice that the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[r, "\tau_A"] & \mathcal{P}\check{\mathcal{P}}A \\
\mathcal{L}_{\mathcal{P}}(\neg) \arrow[u, "{[-]_\alpha}"] \arrow[r, "m_{\{\top\}}"'] & \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg) \arrow[u, "{\mathcal{P}([-]_\alpha)}"']
\end{tikzcd}$$

commutes for every coalgebra α.

Using $m_{\{\top\}}$ we define a syntactic translation $m : \mathcal{L}_{\mathcal{P}}(\Lambda) \longrightarrow \mathcal{L}_{\mathcal{P}}(\neg)$ inductively. To translate formulas of the form $\lambda_{\{\top\}}\varphi$, we use the following diagram

$$\begin{tikzcd}
\mathcal{L}_{\mathcal{P}}(\Lambda) \arrow[r, "m"] \arrow[d, "\Diamond"'] & \mathcal{L}_{\mathcal{P}}(\neg) \arrow[d, dashed] \arrow[dr, "m_{\{\top\}}"] & \\
\mathcal{L}_{\mathcal{P}}(\Lambda) \arrow[r, "m"'] & \mathcal{L}_{\mathcal{P}}(\neg) & \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg) \arrow[l, "\nabla"]
\end{tikzcd}$$

Using this diagram we can describe the translation of a formula $\lambda_{\{\top\}}\varphi$ is

$$m(\lambda_{\{\top\}}\varphi) = \nabla m_{\{\top\}} m(\varphi) = \nabla\{m(\varphi)\}.$$

Using the notation of the blind translator it is $m(\lambda_{\{\top\}}\varphi) = \nabla\{\varphi\}$. It is easy to check that this is in fact a translation.

We will not translate the other singleton liftings in this section. The translation of the other predicate liftings is in the next section on page 81.

Now we believe the reader has enough background and intuition to see the general procedure. Actually we hope that at this point the reader has seen the general procedure by itself.

6.1.3 The general procedure

In this section, we will explain the general procedure used in the previous sections to obtain translations from natural transformations. As stated, the previous section contains two detailed examples of the work done here.

If we want to give an inductive translation from languages of predicate liftings to Moss' languages, then one must realize that the only way to translate a predicate lifting is to use Moss' modality. As we have seen in the examples above and will see in the following pages, in the case of singleton liftings in fact we only have to use Moss' modality.

Our technique to translate a singleton lifting λ, for a functor T, is a three steps technique. The three steps are:

Step 1: Carefully define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow T\check{\mathcal{P}}$.

Step 2: Prove that $\lambda = \nabla\tau$.

Step 3: Extract a translation from τ.

The first two steps are coded in the definition of *natural translator* below. The third step is the hard one and lead us to the concepts of *boolean transformation* and *logical translator*. The third step produces the translation and the first two steps allow us to prove that it is in fact a translation.

Remark 6.1.1. *Notice that the steps of our technique should be executed in the order in which they are presented. But notice that to produce a translation we usually first define the function candidate to be a translation, and then we prove that it is in fact a translation.*

Definition 6.1.2. *A* natural translator *for a singleton lifting λ_p is a natural transformation*

$$\tau : \check{\mathcal{P}} \longrightarrow T\check{\mathcal{P}},$$

such that the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[rr, "\tau_A"] \arrow[dr, "\lambda_p"'] & & T\check{\mathcal{P}}A \arrow[dl, "\nabla_A"] \\
& \check{\mathcal{P}}TA &
\end{tikzcd} \tag{6.1}$$

commutes for every set A. In other words $\nabla\tau = \lambda_p$.

We invite the reader the to check previous section to see two examples of natural translators.

Now we want to obtain a syntactic translation from τ and we want to extend this translation over the inductive steps. For that purposes we will define the concept of *boolean transformation.* A natural transformation $\lambda : F \longrightarrow G$ is a class of arrows indexed over the domain of F and G. A boolean transformation follows the same idea but we do not index over the domain category but over the category of $\mathcal{P}_\kappa + I$-algebras.

Definition 6.1.3 (Boolean transformations)**.** *Let two endofunctors F and G over the category Set be given. A* boolean transformation $\overline{\tau}$ *from F to G (denoted $\overline{\tau} : F \longrightarrow G$)) is a function that assigns to each $\underline{\mathcal{P}_\kappa + I\text{-algebra}}$ $\mathfrak{A} = (A, \wedge, \neg)$ a function $\tau_{\mathfrak{A}} : FA \longrightarrow GA$, an arrow in Set, in such a way that the following naturality condition holds: for each morphism of $\mathcal{P}_\kappa + I$-algebras $f : \mathfrak{A} \longrightarrow \mathfrak{B}$, the square on the right*

$$\begin{tikzcd}
Alg(\mathcal{P}_\kappa + I) & Set & \\
\mathfrak{A} \arrow[d, "f"'] & FA \arrow[r, "\tau_{\mathfrak{A}}"] \arrow[d, "F(f)"'] & GA \arrow[d, "G(f)"] \\
\mathfrak{B} & FB \arrow[r, "\tau_{\mathfrak{B}}"'] & GB
\end{tikzcd}$$

commutes.

Remark 6.1.4. *The reader familiar with category theory should have noticed that boolean transformations are just natural transformations*

$$\overline{\tau} : FU \longrightarrow GU,$$

where U is the forgetful functor $U : Alg(\mathcal{P}_\kappa + I) \longrightarrow Set$.

Remark 6.1.5. *Notice that every natural transformation from F to G is a boolean transformation but not the other way around. We are indexing over the category of $\mathcal{P}_\kappa + I$-algebras and not over the category Set. This has two main consequences: one, we only have a function $\tau_{\mathfrak{A}} : FA \longrightarrow GA$ if the set A is the* *carrier of a $\mathcal{P}_\kappa + I$-algebra. And two, if the set A is the carrier of many different boolean structures, we should have a function from FA to GA for each of these structures.*

We suggest to our reader to have look to the previous section to see two examples of boolean transformations.

The trick to obtain syntactic translations from natural transformations that can be extended over the inductive steps is to use a combination of natural translators and boolean transformations.

Definition 6.1.6. *A natural translator $\tau : \check{\mathcal{P}} \longrightarrow T\check{\mathcal{P}}$ for a singleton lifting λ_p is* logically representable*, or a logical translator, if there exists a* boolean transformation $\overline{\tau} : I \longrightarrow T$ *such that if $\mathfrak{A} = (\check{\mathcal{P}}A, \wedge, \neg)$ is a $\mathcal{P}_\kappa + I$-algebra where $\wedge$ is the A-component of the natural transformation for conjunctions and $\neg$ is the A component of the natural transformation for negations then*

$$\tau_{\mathfrak{A}} = \tau_A.$$

If τ is logically representable, we also say that τ can be extended *to a boolean transformation, or that τ can be extended to $\overline{\tau}$, or that $\overline{\tau}$ extends τ.*

Examples of natural translators that can be extended to boolean transformations can be found in the previous section. For each of the logical translators in the previous section we defined a function

$$m : \mathcal{L}_{\mathcal{P}}(\neg) \longrightarrow \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg),$$

that later we used to produce a translation. The existence of such functions is a general property of logical translators. Later we will illustrate how to use these functions to define a translation.

Lemma 6.1.7. *If $\tau : \check{\mathcal{P}} \longrightarrow T\check{\mathcal{P}}$ is a logical translator for a singleton lifting λ_p then there exists a function $m_p : \mathcal{L}^\kappa_T(\neg) \longrightarrow T\mathcal{L}^\kappa_T(\neg)$ such that the following diagram*

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[r, "\tau_A"] & T\check{\mathcal{P}}A \\
\mathcal{L}^\kappa_T(\neg) \arrow[u, "{[-]_\alpha}"] \arrow[r, "m_p"'] & T\mathcal{L}^\kappa_T(\neg) \arrow[u, "{T([-]_\alpha)}"']
\end{tikzcd} \tag{6.2}$$

commutes for each coalgebra α.

In other words for each $\varphi\mathcal{L}^\kappa_T(\neg)$ the following holds

$$\tau_A([\varphi]_\alpha) = T([m_p(\varphi)]_\alpha)$$

for each coalgebra α.

The function m_p is called the signature *of the translator.*

Proof. Since τ is a logical translator it can be extended to a boolean translation $\overline{\tau} : I \longrightarrow T$. Since $\mathcal{L}_T^\kappa(\neg)$ is a $\mathcal{P}_\kappa + I$-algebra there exists a function

$$\tau_{\mathcal{L}_T^\kappa(\neg)} = m_p : \mathcal{L}_T^\kappa(\neg) \longrightarrow T\mathcal{L}_T^\kappa(\neg).$$

Notice that the satisfaction relation $[-]_\alpha : \mathcal{L}_T^\kappa(\neg) \longrightarrow \check{\mathcal{P}}A$ is a morphism of $\mathcal{P}_\kappa + I$-algebras. Therefore, by definition of boolean translation, the diagram in the statement of the lemma commutes. $\square$

We remark that the domain of the signature of a translator is Moss' language and not the coalgebraic language. As we saw in the previous sections using logical translators we can produce syntactic translations from the language of predicate liftings to the Moss' language. Illustrations of this previous lemma and construction of signatures are discussed in the previous section.

The idea to define a translation from a logical translator is as follows: Moss' language happens to be a $\mathcal{P}_\kappa + T + I$-algebra. To define translation is to define the singleton lifting as a explicit operation of the algebra $(\mathcal{L}_T^\kappa(\neg), \bigwedge, \nabla, \neg)$. Since $\mathcal{L}_T^\kappa(\Lambda)$ is an initial algebra we will give such definition inductively. In other words we will define function

$$? : \mathcal{L}_T^\kappa(\neg) \longrightarrow \mathcal{L}_T^\kappa(\neg)$$

such that the following diagram

$$\begin{tikzcd}
\mathcal{L}_T^\kappa(\Lambda) \arrow[r, "m"] \arrow[d, "\lambda_p"'] & \mathcal{L}_T^\kappa(\neg) \arrow[d, dashed, "?"] \\
\mathcal{L}_T^\kappa(\Lambda) \arrow[r, "m"'] & \mathcal{L}_T^\kappa(\neg)
\end{tikzcd}$$

commutes. The commutativity of this squares represents that our translation is defined inductively.

Theorem 6.1.8. *The predicate lifting λ_p is expressible in the Moss' language if there exists a natural translator τ for λ_p that is logically representable.*

Proof. We use $[-]_{(m,\alpha)}$ for Moss' satisfaction, we use $[-]_{s,\alpha}$ for the satisfaction relation of coalgebraic modal languages.

Since τ is a logical translator, by the previous lemma, there exists a function $m_p : \mathcal{L}_T^\kappa(\neg) \longrightarrow T\mathcal{L}_T^\kappa(\neg)$, the signature for τ, such that for every formula $\psi \in \mathcal{L}_T^\kappa(\neg)$ and every coalgebra α the following holds

$$\tau_A([\psi]_{(m,\alpha)}) = T([m_p(\psi)]). \tag{6.3}$$

We define a syntactic translation $m : \mathcal{L}_T^\kappa(\Lambda) \longrightarrow \mathcal{L}_T^\kappa(\neg)$ inductively. The inductive step to translate formulas of the form $\lambda_p\varphi$ uses the the signature of τ

in the following diagram

$$\begin{tikzcd}
\mathcal{L}^{\kappa}_{T}(\Lambda) \arrow[r, "m"] \arrow[d, "\lambda_p"'] & \mathcal{L}^{\kappa}_{T}(\neg) \arrow[d, dashed, "?"] \arrow[dr, "m_p"] & \\
\mathcal{L}^{\kappa}_{T}(\Lambda) \arrow[r, "m"'] & \mathcal{L}^{\kappa}_{T}(\neg) & T\mathcal{L}^{\kappa}_{T}(\neg). \arrow[l, "\nabla"]
\end{tikzcd}$$

In other words the translation m works as follows

$$m(\lambda_p\varphi) = \nabla m_p m(\varphi).$$

Now we want to see that this is in fact a translation, i.e. we want to prove the following.

Claim 6.1.9. *For every T-coalgebra α and every formula $\varphi \in \mathcal{L}^{\kappa}_{T}(\Lambda)$ the following holds*

$$[\lambda_p\varphi]_{(s,\alpha)} = [\nabla m_p m(\varphi)]_{(m,\alpha)}.$$

We will present a proof of this equality using a chain of equalities that will be extended step by step based on previous results.

Our inductive hypothesis is:

$$[\varphi]_{(s,\alpha)} = [m(\varphi)]_{(m,\alpha)}.$$

In Chapter 4 when we showed how to present coalgebraic languages algebraically we saw that

$$[\lambda_p\varphi]_{(s,\alpha)} = \alpha^{-1}\lambda_{(p,A)}([\varphi]_{(s,\alpha)}).$$

Using the inductive hypothesis it can be extended as follows

$$\begin{aligned}[\lambda_p\varphi]_{(s,\alpha)} &= \alpha^{-1}\lambda_{(p,A)}([\varphi]_{(s,\alpha)})\\ &= \alpha^{-1}\lambda_{(p,A)}([m(\varphi)]_{(m,\alpha)}).\end{aligned}$$

Since τ is a natural translator for λ_p then for every set $X \subseteq A$ we have

$$\alpha^{-1}\lambda_{(p,A)}(X) = \alpha^{-1}\nabla_A\tau_A(X).$$

Replacing X for $[m(\varphi)]_{(m,\alpha)}$ in the previous equation we conclude

$$\alpha^{-1}\lambda_{(p,A)}([m(\varphi)]_{(m,\alpha)}) = \alpha^{-1}\nabla_A\tau_A([m(\varphi)]_{(m,\alpha)}).$$

Using this we can extend our first equality to

$$\begin{aligned}[\lambda_p\varphi]_{(s,\alpha)} &= \alpha^{-1}\lambda_{(p,A)}([\varphi]_{(s,\alpha)})\\ &= \alpha^{-1}\lambda_{(p,A)}([m(\varphi)]_{(m,\alpha)})\\ &= \alpha^{-1}\nabla_A\tau_A([m(\varphi)]_{(m,\alpha)}).\end{aligned}$$

Since $m(\varphi) \in \mathcal{L}^{\kappa}_{T}(\neg)$, we can use equation 6.3, above, and then obtain

$$\alpha^{-1}\nabla_A\tau_A([m(\varphi)]_{(m,\alpha)}) = \alpha^{-1}\nabla_A T([m_p m(\varphi)]_{(m,\alpha)}).$$

With this our chain of equalities is extended to

$$\begin{aligned}[\lambda_p\varphi]_{(s,\alpha)} &= \alpha^{-1}\lambda_{(p,A)}([\varphi]_{(s,\alpha)})\\ &= \alpha^{-1}\lambda_{(p,A)}([m(\varphi)]_{(m,\alpha)})\\ &= \alpha^{-1}\nabla_A\tau_A([m(\varphi)]_{(m,\alpha)})\\ &= \alpha^{-1}\nabla_A T([m_p m(\varphi)]_{(m,\alpha)}).\end{aligned}$$

Finally, recall that in Chapter 3 we saw that using Pattinson transformations we can compute satisfaction sets as follows

$$\alpha^{-1}\nabla_A T([m_p m(\varphi)]_{(m,\alpha)}) = [\nabla m_p m(\varphi)]_{(m,\alpha)}.$$

With all this together our chain becomes.

$$\begin{aligned}[\lambda_p\varphi]_{(s,\alpha)} &= \alpha^{-1}\lambda_{(p,A)}([\varphi]_{(s,\alpha)})\\ &= \alpha^{-1}\lambda_{(p,A)}([m(\varphi)]_{(m,\alpha)})\\ &= \alpha^{-1}\nabla_A\tau_A([m(\varphi)]_{(m,\alpha)})\\ &= \alpha^{-1}\nabla_A T([m_p m(\varphi)]_{(m,\alpha)}) = [\nabla m_p m(\varphi)]_{(m,\alpha)}.\end{aligned}$$

This concludes the proof of the claim.

Therefore we conclude that the function m described above is a translation and this concludes the proof of the theorem. □

Definition 6.1.10. *A translation $m : \mathcal{L}^\kappa_T(\Lambda) \longrightarrow \mathcal{L}^\kappa_T(\neg)$ that is obtained by the procedure described in the previous theorem is said to be* natural for λ_p. *A translation is a* natural translation *if it is natural for all singleton liftings.*

In summary, to produce a natural translation for a singleton lifting λ for a functor T we need three steps. These steps are

Step 1: Define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow T\check{\mathcal{P}}$ that seems like a natural translator for λ.

Step 2: Prove that τ is a natural translator for λ.

Step 3: Show that τ can be extended to a boolean transformation $\overline{\tau} : I \longrightarrow T$

As we saw in Theorem 6.1.8, if we follow these three steps we can produce a natural translation.

Remark 6.1.11. *Technically speaking we produce a natural translation in four steps:*

Step 4: Use the signature of τ to produce a translation for λ.

but as we saw in the light of Theorem 6.1.8 it is enough to do the first three steps and then the fourth one is automatic.

Remark 6.1.12. *Let T' be a functor, λ a singleton lifting for T', and let τ be a logical translator for λ. Now notice the following property of logical translators: For any other functor T Moss' language $\mathcal{L}^\kappa_T(\neg)$ is a $\mathcal{P}_\kappa + I$-algebra. Since τ is* *a logical translator it can be extended to a boolean transformation* $\overline{\tau} : I \longrightarrow T'$*, then there exists a function*

$$\tau_{\mathcal{L}^{\kappa}_{T}(\neg)} : \mathcal{L}^{\kappa}_{T}(\neg) \longrightarrow T'\mathcal{L}^{\kappa}_{T}(\neg).$$

The existence of these functions is what will help us to extend logical translators over the inductive steps. For example, using the coproduct inclusions we obtain an arrow

$$i_T \tau_{\mathcal{L}_{T+T'}(\neg)} : \mathcal{L}_{T+T'}(\neg) \longrightarrow (T\mathcal{L}_{T+T'}(\neg) + T'\mathcal{L}_{T+T'}(\neg)).$$

In later sections we will see that this arrow will be the signature of a natural translator for the predicate lifting $(\lambda, \lambda_{\perp}) : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}(T + T')$*.*

Some reasons why logical translators will be useful because are the following:

1. Natural translators help us to describe Pattinson transformations using predicate liftings.
2. Since every logical translator can be extended to a boolean transformation they produce syntactic translations.
3. We can appropriately redefine logical translators over the inductive steps, using technics from category theory, in such a way that we obtain new logical translators.

Before showing how to extend logical translators over all the inductive steps we will produce natural translations for the other remaining base cases, i.e. the constant functors and the homomorphism functors.

Power set functor (continued)

First we will define logical translators for the remaining singleton liftings for the covariant power set functor. We will only mention which is the natural translator and what is its signature. The construction of the other components is clear from that.

We can construct a logical translator for the singleton lifting associated with $\emptyset$ using:

- A natural translator $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$. The components of τ are the constant function with value $\emptyset$.
- The signature of τ is the function $m_{\{\emptyset\}} : \mathcal{L}_{\mathcal{P}}(\neg) \longrightarrow \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg)$ with constant value $\emptyset$.
- The translation is
$$m(\lambda_{\{\emptyset\}}\varphi) = \nabla\emptyset$$

We can construct a logical translator for the singleton lifting associated with $\{\perp\}$ using:

- A natural translator $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$ with the following components: The A-component maps a set $X \subseteq A$ to $\{\neg X\}$
- The signature of τ is the function $m_{\{\perp\}} : \mathcal{L}_{\mathcal{P}}(\neg) \longrightarrow \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg)$ which maps a formula φ to $\{\neg\varphi\}$.
- The translation is

$$m(\lambda_{\{\perp\}}\varphi) = \nabla\{\neg m(\varphi)\}$$

We can construct a logical translator for the singleton lifting associated with $\{\perp, \top\}$ using:

- A natural translator $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$ with the following components: The A-component maps a set $X \subseteq A$ to $\{\neg X, X\}$
- The signature of τ is the function $m_{\{\perp\}} : \mathcal{L}_{\mathcal{P}}(\neg) \longrightarrow \mathcal{P}\mathcal{L}_{\mathcal{P}}(\neg)$ which maps a formula φ to $\{\neg\varphi, \varphi\}$.
- The translation is

$$m(\lambda_{\{\perp,\top\}}\varphi) = \nabla\{\neg m(\varphi), m(\varphi)\}$$

Summarizing we have that a translation for the covariant set functor is a function $m : \mathcal{L}^{\kappa}_{T}(\Lambda) \longrightarrow \mathcal{L}^{\kappa}_{T}(\neg)$ defined as follows:

$$\begin{aligned} m([\lambda_{\{\emptyset\}}]\varphi) &= \nabla\emptyset. \\ m([\lambda_{\{\top\}}]\varphi) &= \nabla\{m(\varphi)\}. \\ m([\lambda_{\{\perp\}}]\varphi) &= \nabla\{m(\neg\varphi)\}. \\ m([\lambda_{\{\top,\perp\}}]\varphi) &= \nabla\{m(\varphi), \neg m(\varphi)\}. \end{aligned}$$

6.1.4 Constant functors

In this section we assume we are working with a constant functor with value D. A coalgebra for a constant functor is a function $\alpha : A \longrightarrow D$. Moss' Logic for this functor can be found on page 27. Coalgebraic modal logic for this functor can be found on page 46.

The predicate lifting associated with $d \in D$ is the constant function with value d. The natural translator $\tau : \check{\mathcal{P}} \longrightarrow D$ associated with λ_d has as components constant functions with value d. Clearly this translator can be extended to a boolean transformation. Hence it is a logical translator. The signature $m_p : \mathcal{L}_D(\neg) \longrightarrow D$ of this translator is the constant function with value d. From all this we conclude that in the case of constant functors the natural translation is as follows:

Using Theorem 6.1.8 we define $m : \mathcal{L}_D(\Lambda) \longrightarrow \mathcal{L}_D(\neg)$ inductively as follows:

$$\begin{aligned} m([\lambda_d]\varphi) &= \nabla d \\ m([\lambda_C]\varphi) &= \bigvee_{d \in C} \nabla d. \end{aligned}$$

6.1.5 $Hom(D,-)$ functors

A coalgebra for this functor is a function $\alpha : A \longrightarrow Hom(D,A)$. Moss Logic for this functor can be found on page 27. Coalgebraic modal logic for this functor can be found on page 47.

As we said before, a singleton lifting for this functor is associated with $P \subseteq D$. Now we will define a natural translator $\tau : \check{\mathcal{P}} \longrightarrow Hom(D, \check{\mathcal{P}}(-))$using this set P. The component $\tau_A : \check{\mathcal{P}}A \longrightarrow Hom(D, \check{\mathcal{P}}A)$ of τ acts as follows: A set $X \subseteq A$ is mapped to

$$\begin{aligned} X_P : D &\longrightarrow \check{\mathcal{P}}A \\ d &\longmapsto \begin{cases} X \text{ if } d \in P, \\ \neg X \text{ if } d \notin P \end{cases} \end{aligned}$$

where $\neg X$ is the complement of X. Now we will show that this is in fact a natural transformation. We want to show that for a function $g : A \longrightarrow B$ and for every $X \subseteq A$,

$$g^{-1}(X_P) = g^{-1}(X)_P.$$

By definition we have

$$\begin{aligned} g^{-1}X_P : D &\longrightarrow B \\ d &\longmapsto \begin{cases} g^{-1}(X) \text{ if } d \in P \\ g^{-1}(\neg X) \text{ if } d \notin P \end{cases} \end{aligned}$$

Since the inverse image of g commutes with complements, we conclude the desired equality.

In the previous construction we only used that g^{-1} is a morphism of $\mathcal{P}_\kappa + I$-algebras. From that we can see that we can extend τ to a boolean transformation $\overline{\tau} : I \longrightarrow Hom(D,-)$ as follows: Let $\mathfrak{A}$ be a $\mathcal{P}_\kappa + I$-algebra and $x \in A$ an element in the carrier. The element x is mapped to

$$\begin{aligned} x_P : D &\longrightarrow \mathfrak{A} \\ d &\longmapsto \begin{cases} x \text{ if } d \in P, \\ \neg x \text{ if } d \notin P \end{cases} \end{aligned}$$

where $\neg x$ is the complement of x in $\mathfrak{A}$.

The signature of τ is: $m_P : \mathcal{L}_T^\kappa(\neg) \longrightarrow Hom(D, \mathcal{L}_T^\kappa(\neg))$, where $T = Hom(D,-)$. A formula φ is mapped to $\varphi_P : D \longrightarrow \mathcal{L}_T^\kappa(\neg)$, this function is defined as follows:

$$\varphi_P(d) = \varphi \text{ if } d \in P; \varphi_P(d) = \neg\varphi \text{ if } d \notin P.$$

Using this signature we define a natural translation $m : \mathcal{L}_T^\kappa(\Lambda) \longrightarrow \mathcal{L}_T^\kappa(\neg)$ following the idea of Theorem 6.1.8. This translation acts on predicate liftings as follows:

$$m([\lambda_P]\varphi) = \nabla m(\varphi)_P,$$

With this we finish all the base cases. Now we proceed to translate the inductive steps.

6.1.6 Coproducts of functors

Before explaining the general case, we will use the example $\mathcal{P} + I$ to show how to extend natural translations of the base cases to coproducts.

Recall that a predicate lifting for a coproduct is the product of two predicate liftings, one for each of the factors. Also recall that a singleton lifting is associated with an element p in one of the factors of the coproduct.

Let's consider the predicate lifting associated wit $\bot \in 2$. The associated predicate lifting for the coproduct is $([\neg], \lambda_\bot)$. Recall that this predicate lifting maps $X \subseteq A$ to $\neg X \subseteq A + \mathcal{P}A$. Now notice that the translator $\neg : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}$ can be extended to $\neg : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}} + \mathcal{P}\check{\mathcal{P}}$ using the appropriate injections. Furthermore notice that this extension is a natural translator for $([\neg], \lambda_\bot)$. The signature $m_\bot : \mathcal{L}^\kappa_T(\neg) \longrightarrow (\mathcal{L}^\kappa_T(\neg) + \mathcal{P}\mathcal{L}^\kappa_T(\neg))$ maps a formula $\varphi \in \mathcal{L}^\kappa_T(\neg)$ to $\neg\varphi$. Explicitly the translation is

$$m(\lambda_\bot \varphi) = \nabla(\neg\varphi),$$

where nabla is now the Moss' modality for the coproduct.

Lets consider another example before explaining the general case. We consider the case $\{\top\} \in \mathcal{P}2$. The associated predicate lifting is $(\lambda_\bot, \lambda_{\{\top\}})$. Again we have a natural translator $\tau : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}}$. We extend it to $\tau : \check{\mathcal{P}} \longrightarrow (\check{\mathcal{P}} + \mathcal{P}\check{\mathcal{P}})$ using the appropriate injections. Again this is a natural translator for $(\lambda_\bot, \lambda_{\{\top\}})$. The signature $m_{\{\top\}} : \mathcal{L}^\kappa_T(\neg) \longrightarrow (\mathcal{L}^\kappa_T(\neg) + \mathcal{P}\mathcal{L}^\kappa_T(\neg))$ maps a formula $\varphi \in \mathcal{L}^\kappa_T(\neg)$ to $\{\varphi\}$. Explicitly the translation is

$$m(\lambda_{\{\top\}}\varphi) = \nabla\{m(\varphi)\},$$

where nabla is now the Moss' modality for the coproduct.

The general case

Moss' Logic for coproducts can be found on page 29. Coalgebraic modal logic for this functor can be found on page 48. A coalgebra for the coproduct is a function $\alpha : A \longrightarrow K_1 A + K_2 A$. Now we explain the general procedure.

Given $p \in K_1 2 + K_2 2$ we want to translate the predicate lifting

$$\lambda_{(p,12)} : 2^{(-)} \longrightarrow 2^{K_1 + K_2}$$

into the Moss' language for $K_1 + K_2$. It is our claim that we can obtain such translation using logical translators for K_1 and K_2. Furthermore we claim that we can obtain a natural translation, i.e. a translation constructed from a logical translator as described in Theorem 6.1.8. In other words we want to prove the following theorem.

Theorem 6.1.13. *Let K_1 and K_2 be two functors such that:*

- *We can define Moss' language for both of them.*
- *Every singleton lifting for K_1 can be translated into the Moss' language of K_1 using logical translators.*

- *Every singleton lifting for K_2 can be translated into the Moss' language of K_2 using logical translators.*

Then every singleton lifting for K_1+K_2 can be translated into the Moss' language for $K_1 + K_2$ using logical translators.

Assume $p \in K_1 2$. Recall that the predicate lifting $\lambda_{(p,12)}$ is a pair $(\lambda_p, \lambda_\perp)$ of predicate liftings, where λ_p is the predicate lifting $\lambda_p : 2^{(-)} \longrightarrow 2^{K_1(-)}$ and $\lambda_\perp$ is the predicate lifting associated with the empty set for K_2. In order to produce a logical translator for $(\lambda_p, \lambda_\perp)$ we have to do three things:

1. Define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}} + K_2\check{\mathcal{P}}$ that has a chance of being a natural translator for $(\lambda_p, \lambda_\perp)$.
2. Prove that τ is a natural translator for $(\lambda_p, \lambda_\perp)$.
3. Show that τ can be extended to a boolean transformation $\overline{\tau} : I \longrightarrow K_1+K_2$.

Our inductive hypothesis is: For each $p \in K_1 2$ there exists a natural translator

$$\tau_p : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}},$$

that is logically representable, i.e. τ produces a natural translation for $\lambda_p : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}}$. The same holds for each $p \in K_2 2$.

First we define τ. By inductive hypothesis there exists a logical translator

$$\tau_p : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}}$$

for $\lambda_p : 2^{(-)} \longrightarrow 2^{K_1(-)}$. By definition of coproducts there exists a natural transformation

$$i_{K_1} : K_1\check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}} + K_2\check{\mathcal{P}}.$$

Composing these two natural transformations we obtain a natural transformation

$$\tau = i_{K_1}\tau_p : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}} + K_2\check{\mathcal{P}}.$$

Claim 6.1.14. *The natural transformation τ, as defined above, is a natural translator for $(\lambda_p, \lambda_\perp)$.*

In order to prove this claim we have to show that the following diagram

$$\begin{tikzcd} \mathcal{P}A \arrow[rr, "(i_{K_1}\tau_p)_A"] \arrow[dr, "(\lambda_p,\lambda_\perp)_A"'] & & K_1\mathcal{P}A + K_2\mathcal{P}A \arrow[dl, "\nabla_A"] \\ & \mathcal{P}A & \end{tikzcd} \tag{6.4}$$

commutes for each set A. In other words we want to proof $(\lambda_p, \lambda_\perp) = \nabla\tau$.

Since τ_p is a translator for λ_p, then for each $X \subseteq A$,

$$\lambda_{(p,A)}(X) = \nabla_{1,A}\tau_{(p,A)}(X),$$

where ∇_1 is the Pattinson transformation for K_1. From now on we will omit the subindex A.

This previous equality is saying the following

$$\lambda_p(X) = \{s \in K_1A \mid (s, \tau_p(X)) \in \overline{K_1}(\in_A)\}. \tag{6.5}$$

The commutativity of triangle 6.4 will follow once we prove that

$$(\lambda_p, \lambda_\perp)(X) = \{s \in (K_1A + K_2A) \mid (s, i_1\tau_p(X)) \in \overline{K_1 + K_2}(\in_A)\},$$

where $i_1 = i_{K_1}$. This follows gathering the next two facts: one, by definition, see page 48, we have

$$(\lambda_p, \lambda_\perp)(X) = \{s \in K_1A \mid s \in \lambda_p(X)\}.$$

Two, the relation lifting for the coproduct is

$$\overline{K_1 + K_2}(\in_A) = \{(i_1(s), i_1(X)) \mid (s, X) \in \overline{K_1(\in_A)}\} \cup \{(i_2(s), i_2(X)) \mid (s, X) \in K_2(\in_A)\}.$$

Then Equation (6.5), above, implies the desired equality. This concludes the proof of the claim. Therefore we conclude that τ is a natural translator for $(\lambda_p, \lambda_\perp)$.

Now we want to obtain a syntactic translation from τ, i.e. we want to show that the new translator $\tau = i_{K_1}\tau_p$ is logically representable. Since τ_p is logically representable there exist a boolean transformation

$$\overline{\tau_p} : I \longrightarrow K_1$$

that extends it. Notice that the injection $i_{K_1} : K_1 \longrightarrow K_1 + K_2$ is a boolean transformation because it is a natural transformation. Composing these two boolean transformations we obtain a boolean transformation

$$i_{K_1}\overline{\tau_p} : I \longrightarrow (K_1 + K_2).$$

Recall that the composition of boolean transformations is again a boolean transformation. Since $\overline{\tau_p}$ extends τ_p it follows that $i_{K_1}\overline{\tau_p}$ also extends $\tau = i_{K_1}\tau_p$. We conclude that τ is not only a translator for $(\lambda_p, \lambda_\perp)$ but a logical translator. The case $p \in K_2 2$ is symmetric. This concludes the proof of the theorem.

Corollary 6.1.15. *Under the assumptions of the previous theorem, if we allow $K_1 2 + K_2$ conjunctions in the Moss' language, then every monadic singleton lifting for $K_1 + K_1$ can be translated into the Moss' language for $K_1 + K_2$.*

Proof. We translate singleton liftings using the previous theorem. Using Theorem 4.3.6, we translate the other predicate liftings. □

An informal description of the natural translation obtained above will be as follows. For $p \in K_1 2$, the signature of τ is the function

$$m_1 = (i_{K_1}\tau_p)_{\mathcal{L}_T^\kappa(\neg)} : \mathcal{L}_T^\kappa(\neg) \longrightarrow (K_1\mathcal{L}_T^\kappa(\neg) + K_2\mathcal{L}_T^\kappa(\neg)),$$

where $T = K_1 + K_2$. Explicitly, the translation $m : \mathcal{L}^{\kappa}_T(\Lambda) \longrightarrow \mathcal{L}^{\kappa}_T(\neg)$ maps a formula $[\lambda_p, \lambda_\perp]\varphi$ to

$$m([\lambda_p, \lambda_\perp]\varphi) = \nabla m_1(\varphi).$$

The case $p \in K_2 2$ is symmetric.

Corollary 6.1.16. *Under the assumptions of the previous theorem, if $K_1 2$ and $K_2 2$ are finite, then every predicate lifting for $K_1 + K_2$ can be translated into Moss' language for $K_1 + K_2$ using a finitary translation.*

Corollary 6.1.17. *If K_1 and K_2 are Kripke polynomial functors, then every predicate lifting for $K_1 + K_2$ can be translated into Moss' language for $K_1 + K_2$. If both functors are finitary, we obtain a finitary translation.*

6.1.7 Product of functors

First we will use the example $\mathcal{P} \times D$ to show how to extend the natural translations of the base cases to products.

We will only consider the singleton lifting associated with $(d, \{\perp\}) \in D \times \mathcal{P}(2)$. Recall that the A-component of this predicate lifting maps a set $X \subseteq A$ to

$$\lambda_d(X) \times \lambda_{\{\perp\}}(X) = \{d\} \times \{\neg X\}.$$

We know that there exist logical translators,

$$\tau_d : \check{\mathcal{P}} \longrightarrow D, \quad \tau_{\{\perp\}} : \check{\mathcal{P}} \longrightarrow \mathcal{P}\check{\mathcal{P}},$$

associated with λ_d and $\lambda_{\{\perp\}}$ respectively. Since we are working with a product functor, the only reasonable attempt to construct a natural translator is to use the universal property of the product to obtain a new translator

$$(\tau_d, \tau_{\{\perp\}}) : \check{\mathcal{P}} \longrightarrow D \times \mathcal{P}\check{\mathcal{P}}.$$

Clearly it is again a natural transformation and doing a simple computation we can see that in fact it is a natural translator for $\lambda_{(d,\{\perp\})}$. The signature of this translator, $m_{(d,\{\perp\})} : \mathcal{L}^{\kappa}_T(\neg) \longrightarrow D \times \mathcal{P}\mathcal{L}^{\kappa}_T(\neg)$, maps a formula $\varphi \in \mathcal{L}^{\kappa}_T(\neg)$ to $(d, \{\neg\varphi\})$. From that we can see that a translation is

$$m(\lambda_{(d,\{\perp\})}\varphi) = \nabla(d, \{\neg\varphi\}).$$

The general case

Let a pair of functors $K_1, K_2 : Set \longrightarrow Set$ be given. In this section we will investigate Moss language and the Coalgebraic modal language for the product functor $K_1 \times K_2 : Set \longrightarrow Set$.

Moss Logic for this functor can be found on page 30. Coalgebraic modal logic for this functor can be found on page 49.

A singleton predicate lifting for the product is associated with a pair $(p_1, p_2) \in K_1 2 \times K_2 2$. The A-component of this predicate lifting is a function

$$\lambda_{(p_1,p_2)} : 2^A \longrightarrow 2^{K_1 A \times K_2 A}.$$

Such function maps a set $X \subseteq A$ to $\lambda_{p_1}(X) \times \lambda_{p_2}(X)$. We want to translate $\lambda_{(p_1,p_2)}$ into the Moss' language for $K_1 \times K_2$. It is our claim that we can do it using a natural translation. In order to show that, we will prove the following result.

Theorem 6.1.18. *Let K_1 and K_2 be two functors such that:*

- *We can define Moss' language for both of them.*
- *Every singleton lifting for K_1 can be translated into the Moss' language of K_1 using logical translators.*
- *Every singleton lifting for K_2 can be translated into the Moss' language of K_2 using logical translators.*

Then every singleton lifting for $K_1 \times K_2$ can be translated into the Moss' language for $K_1 \times K_2$ using logical translators.

Fix $(p_1, p_2) \in K_1 2 \times K_2 2$. To produce a logical translator for $\lambda_{(p_1,p_2)}$ we have to do three things:

1. Define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}} \times K_2\check{\mathcal{P}}$ that will be a natural translator for $\lambda_{(p_1,p_2)}$.
2. Prove that τ is a natural translator for $\lambda_{(p_1,p_2)}$.
3. Show that τ can be extended to a boolean transformation $\overline{\tau} : I \longrightarrow K_1 \times K_2$

Our inductive hypothesis is: for each $p_1 \in K_1 2$ there exists a natural translator

$$\tau_{p_1} : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}}$$

that is logically representable. The same holds for each $p_2 \in K_2 2$.

First we define τ. By inductive hypothesis there exists logical translators

$$\tau_{p_1} : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}}; \quad \tau_{p_2} : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}}$$

for λ_{p_1} and λ_{p_2} respectively. By the universal property of the product there exists a natural transformation

$$\tau = (\tau_{p_1}, \tau_{p_2}) : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}} \times K_2\check{\mathcal{P}}.$$

In other words, the natural translator associated with $\lambda_{(p_1,p_2)}$ is the product of τ_{p_1} and τ_{p_2}. Recall that the A-component of τ maps a set $X \subseteq A$ to $(\tau_{(p_1,A)}(X), \tau_{(p_2,A)}(X))$.

Now we want to show that τ is a natural translator for $\lambda_{(p_1,p_2)}$, i.e. we want to show

$$\lambda_{(p_1,p_2)}(X) = \{(s_1, s_2) \in K_1A \times K_2A \,|\, ((s_1, s_2), (\tau_{p_1}, \tau_{p_2})(X)) \in \overline{(K_1 \times K_2)}(\in_A)\}. \tag{6.6}$$

By induction hypotheses we have

$$\lambda_{p_1}(X) = \{s_1 \in K_1A \,|\, (s_1, \tau_{p_1}(X)) \in \overline{K_1}(\in_A)\} \text{ and}$$
$$\lambda_{p_2}(X) = \{s_2 \in K_2A \,|\, (s_2, \tau_{p_1}(X)) \in \overline{K_2}(\in_A)\}.$$

The relation lifting of the membership relation with a product is:

$$\overline{K_1 \times K_2}(\in_A) = \{((x_0, x_1), (x_0', x_1')) \mid (x_0, x_0') \in \overline{K_1}(\in_A) \text{ and } (x_1, x_1') \in \overline{K_2}(\in_A)\}.$$

Putting this and the inductive hypothesis together we can prove Equation 6.6 after a straight forward computation. We conclude that τ is a logical translator for $\lambda_{(p_1,p_2)}$.

Now we have to show that $(\tau_{p_1}, \tau_{p_2}) : \check{\mathcal{P}} \longrightarrow K_1\check{\mathcal{P}} \times K_2\check{\mathcal{P}}$ is logically representable. Let $\mathfrak{A}$ be a $\mathcal{P}_\kappa + I$-algebra with carrier set A. Since τ_{p_1} and τ_{p_1} can be extended to boolean translations there exists functions

$$\tau_{(p_1,\mathfrak{A})} : A \longrightarrow K_1A \text{ and } \tau_{(p_2,\mathfrak{A})} : A \longrightarrow K_2A.$$

Using the universal property of products we obtain a function

$$(\tau_{(p_1,\mathfrak{A})}, \tau_{(p_1,\mathfrak{A})}) : A \longrightarrow K_1A \times K_2A.$$

It is easy to see that these are the components of a boolean translation

$$\overline{(\tau_{p_1}, \tau_{p_2})} : I \longrightarrow K_1 + K_2$$

extending the natural translator (τ_{p_1}, τ_{p_2}). Notice that $\overline{(\tau_{p_1}, \tau_{p_2})}$ is the product of $\overline{\tau_{p_1}}$ and $\overline{\tau_{p_2}}$. We conclude that (τ_{p_1}, τ_{p_2}) is a logical translator for $\lambda_{(p_1,p_2)}$. With this we conclude the proof of the theorem.

Corollary 6.1.19. *Under the assumptions of the previous theorem, if we allow $K_1 2 \times K_2 2$ conjuntions in Moss' language, then every predicate lifting for $K_1 \times K_1$ can be translated into the Moss' language for $K_1 \times K_2$.*

Proof. We translate singleton liftings using the previous theorem. Using Theorem 4.3.6 we translate the other predicate liftings. □

An informal description of the natural translation is as follows: Put $T = K_1 \times K_2$, let $m_1 : \mathcal{L}^\kappa_T(\neg) \longrightarrow K_1\mathcal{L}^\kappa_T(\neg)$ be the $\mathcal{L}^\kappa_T(\neg)$-component of $\overline{\tau_{p_1}}$, and let $m_2 : \mathcal{L}^\kappa_T(\neg) \longrightarrow K_2\mathcal{L}^\kappa_T(\neg)$ be the $\mathcal{L}^\kappa_T(\neg)$ component of $\overline{\tau_{p_2}}$. The signature of (τ_{p_1}, τ_{p_2}) is the product function of m_1 and m_2,

$$(m_1, m_2) : \mathcal{L}^\kappa_T(\neg) \longrightarrow K_1\mathcal{L}^\kappa_T(\neg) \times K_2\mathcal{L}^\kappa_T(\neg),$$

A formula $[\lambda_{(p_1,p_2)}]\varphi$ is translated into

$$m([\lambda_{(p_1,p_2)}]\varphi) = \nabla(m_1(\varphi), m_2(\varphi))$$

Corollary 6.1.20. *Under the assumptions of the previous theorem, if $K_1 2$ and $K_2 2$ are finite, then every predicate lifting for $K_1 \times K_2$ can be translated into Moss' language for $K_1 + K_2$ using a finitary translation.*

Corollary 6.1.21. *If K_1 and K_2 are Kripke polynomial functors, then every predicate lifting for $K_1 \times K_2$ can be translated into Moss' language for $K_1 \times K_2$. If both functors are finitary we obtain a finitary translation.*

$Hom(D, K)$ functors are a particular case of products, hence we do not explain their translations.

6.1.8 $\mathcal{P}K$ Functors

A coalgebra for this functor is a function $\alpha : A \longrightarrow \mathcal{P}KA$. Moss Logic for this functor can be found on page 31. Coalgebraic modal logic for this functor can be found on page 50.

Consider the predicate lifting associated with $\{\bot, \top\} \in \mathcal{P}2$ for the covariant power set functor. As we saw above, the translation for this predicate lifting is:

$$m(\lambda_{\{\top,\bot\}}\varphi) = \nabla\{m(\varphi), \neg m(\varphi)\}.$$

Notice that we can restate it as follows

$$m(\lambda_{\{\top,\bot\}}\varphi) = \nabla\{m_\top m(\varphi), m_\bot m(\varphi)\},$$

where $m_\top$, and $m_\bot$ are the signatures for the natural translators of the predicate liftings [1] and [$\neg$] for the identity functor. Furthermore notice that this idea also works for the natural translator, i.e the A-component for a natural translator τ for $\lambda_{\{\bot,\top\}}$ maps a set $X \subseteq A$ ro

$$\tau_A(X) = \{\tau_{A,\top}(X), \tau_{A,\bot}(X)\},$$

where $\tau_\top$ and $\tau_\bot$ are the natural translators for [1] and [$\neg$], respectively.

The general procedure

Let $\lambda_P : \check{\mathcal{P}} \longrightarrow \mathcal{P}K\check{\mathcal{P}}$ be a singleton lifting for PK associated with a set $P \subseteq K2$. We want to translate Λ_P into the Moss' language for $\mathcal{P}K$. One more time we claim that we can in fact obtain a natural translation. In other words, we want to prove the following result.

Theorem 6.1.22. *Let K be a functor such that:*

- *We can define Moss' language for K.*
- *Every singleton lifting for K can be translated into the Moss' language of K using logical translators.*

Then every singleton lifting for $\mathcal{P}K$ can be translated into the Moss' language for $\mathcal{P}K$ using logical translators.

Fix $P \subseteq K2$. To produce a logical translator for λ_P we have to do three things:

1. Define a natural transformation $\tau_P : \check{\mathcal{P}} \longrightarrow \mathcal{P}K\check{\mathcal{P}}$ that will be a natural translator for λ_P.
2. Prove that τ_P is a natural translator for λ_P.
3. Show that τ_P can be extended to a boolean transformation $\overline{\tau} : I \longrightarrow \mathcal{P}K$

Our inductive hypothesis is: for each $p \in K2$ there exists a logical translator

$$\tau_p : \check{\mathcal{P}} \longrightarrow K\check{\mathcal{P}}$$

for $\lambda_p : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}K$.

As always we first define τ_P. By inductive hypothesis for each $p \in P$ there exists a logical translator

$$\tau_p : \check{\mathcal{P}} \longrightarrow K\check{\mathcal{P}}$$

for $\lambda_p : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}K$. Since each τ_p is a logical translator, it is a natural transformation. We define a natural translator

$$\tau_P : \check{\mathcal{P}} \longrightarrow \mathcal{P}K\check{\mathcal{P}}$$

for λ_P as follows: The A-component of τ_P maps a set $X \subseteq A$ to

$$\tau_{(P,A)}(X) = \{\tau_{(p,A)}(X) \,|\, p \in P\},$$

where τ_p is a logical translator for λ_p. Since all τ_p are natural transformations it is straight forward to check that τ_P is a natural transformation.

Now we have to prove that τ_P is in fact the right translator, i.e we have to show that the following diagram

$$\begin{tikzcd}
\check{\mathcal{P}}A \arrow[rr, "\tau_P"] \arrow[dr, "\lambda_P"'] & & \mathcal{P}K\check{\mathcal{P}}A \arrow[dl, "\nabla_A"] \\
& \check{\mathcal{P}}KA &
\end{tikzcd}$$

commutes for every set A, where ∇ is the Moss' modality for $\mathcal{P}K$. From now on we will omit the subindex A. By induction hypothesis we have that for each $p \in K2$ the following holds

$$\lambda_p(X) = \{s \in KA \,|\, (s, \tau_p(X)) \in \overline{K}\,(\in_A)\}.$$

In Section 4.4.8 of Chapter 4, we saw that $\lambda_P : \check{\mathcal{P}} \longrightarrow \check{\mathcal{P}}\mathcal{P}K$ has the following characterization

$$\lambda_P(X) = \{U \subseteq KA \,|\, U \subseteq \bigcup_{p \in P} \lambda_p(X) \wedge (\forall p \in P)(U \cap \lambda_p(X) \neq \emptyset)\}.$$

In Section 3.3.8 of Chapter 3, we saw that Pattinson transformation for $\mathcal{P}K$ can be described as follows: A set $\tau_P(X) \subseteq K\check{\mathcal{P}}A$ is mapped to

$$\begin{aligned}\nabla_A(\tau_P(X)) = \big\{U \subseteq KA \,|\, &(\forall q \in \tau_P(X))(\exists (u \in U))((q,u) \in \overline{K}(\in_A)) \\ &\text{and } (\forall u \in U)(\exists (q \in \tau_P(X)))((q,u) \in \overline{K}(\in_A))\big\}.\end{aligned}$$

Using the inductive hypothesis we can see that

$$(\forall u \in U)(\exists (q \in \tau_P(X)))((q,u) \in \overline{K}(\in_A))$$

is equivalent to

$$U \subseteq \bigcup_{p \in P} \lambda_p(X).$$

Using the inductive hypothesis we can also see that

$$(\forall q \in \tau_P(X))(\exists (u \in U))((q,u) \in \overline{K}(\in_A))$$

is equivalent to

$$(\forall p \in P)(U \cap \lambda_p(X) \neq \emptyset).$$

Therefore we conclude that τ_P is a natural translator for λ_P.

Now we will show that τ_P is logically representable. Let $\mathfrak{A}$ be a $\mathcal{P}_\kappa + I$-algebra over a set A. By induction hypothesis all the translators $\tau_p : \check{\mathcal{P}} \longrightarrow K\check{\mathcal{P}}$ can be extended to boolean transformations. Therefore for each $p \in P$ there exists a function

$$\tau_{(p,\mathfrak{A})} : A \longrightarrow KA.$$

Using these boolean transformations we define a boolean transformation

$$\overline{\tau_P} : \check{\mathcal{P}} \longrightarrow \mathcal{P}K\check{\mathcal{P}}$$

as follows: the $\mathfrak{A}$-component of $\overline{\tau_P} : \check{\mathcal{P}} \longrightarrow \mathcal{P}K\check{\mathcal{P}}$ maps an element $x \in A$ to

$$\tau_{(P,\mathfrak{A})}(x) = \{\tau_{(p,\mathfrak{A})}(x) \mid p \in P\}.$$

It is straightforward to check that these are the components of a boolean translation $\overline{\tau_P} : I \longrightarrow \mathcal{P}K$ extending τ_P. We conclude that τ_P is logical translator for λ_P. This concludes the proof of the theorem.

Corollary 6.1.23. *Under the assumptions of the previous theorem, if we allow $\mathcal{P}K2$ conjunctions in Moss' language, then every predicate lifting for $\mathcal{P}K$ can be translated into the Moss' language for $\mathcal{P}K$.*

Proof. Using Theorem 6.1.8 we produce a natural translation for λ_P. Using Theorem 4.3.6 we translate the other predicate liftings. □

The signature $m_P : \mathcal{L}^\kappa_T(\neg) \longrightarrow \mathcal{P}T\mathcal{L}^\kappa_T(\neg)$, where $T = \mathcal{P}K$, of τ_P maps a formula $\varphi \in \mathcal{L}^\kappa_T(\neg)$ to the set

$$m_P(\varphi) = \{m_p(\varphi) \mid p \in P\},$$

where $m_p : \mathcal{L}^\kappa_T(\neg) \longrightarrow \mathcal{P}K\mathcal{L}^\kappa_T(\neg)$ are the $\mathcal{L}^\kappa_T(\neg)$-components of the boolean transformations associated with the singleton liftings of K. In other words the the translation is the following

$$m(\lambda_P\varphi) = \nabla\{m_p m(\varphi) \mid p \in P\}.$$

Corollary 6.1.24. *Under the assumptions of the previous theorem, if $K2$ is finite, then every predicate lifting for $\mathcal{P}K$ can be translated into Moss' language for $\mathcal{P}K$ using a finitary translation.*

Corollary 6.1.25. *If K is a Kripke polynomial functors, then every predicate lifting for $\mathcal{P}K$ can be translated into Moss' language for $\mathcal{P}K$. If K is finitary we obtain a finitary translation.*

Chapter 7

Conclusions

In this section we summarize the results obtained in this thesis and mention our ideas about further work.

What we have done

Based on the work done in Chapters 2, 3, 4, we proved the following result.

Theorem. *For every κ-accessible functor T that preserves weak pullbacks and monomorphisms there exists a separating set of predicate liftings Λ such that Moss' language $\mathcal{L}^{\kappa}_{T}(\neg)$ and the coalgebraic modal language $\mathcal{L}^{\kappa}_{T}(\Lambda)$ are indistinguishable at a semantic level.*

In Chapter 5 we proved the following result.

Theorem. *The existence of an expressive language for T-coalgebras is equivalent to the existence of a final T-coalgebra.*

Using this construction we defined translations and proved the following theorem.

Theorem. *For every regular cardinal κ and every functor T that preserves weak pullbacks and monomorphisms there exists a separating set of predicate liftings Λ such that:*

- *Every formula in $\mathcal{L}^{\kappa}_{T_\kappa}(\neg)$ can be translated into a formula in $\mathcal{L}_{T_\kappa}(\Lambda)$.*
- *Every formula in $\mathcal{L}^{\kappa}_{T_\kappa}(\Lambda)$ can be translated into a formula in $\mathcal{L}_{T_\kappa}(\neg)$.*

In Chapter 6 we defined the concepts of *natural translator, logical translator and boolean transformation.* In that chapter we also developed a three steps technique to produce logical translators. Recall that our technique is:

Step 1: Define a natural transformation $\tau : \check{\mathcal{P}} \longrightarrow T\check{\mathcal{P}}$ that seems like a natural translator for λ.

Step 2: Prove that τ is a natural translator for λ.

Step 3: Show that τ can be extended to a boolean transformation $\overline{\tau} : I \longrightarrow T$

Using this procedure we proved the following result.

Theorem. *A singleton predicate lifting λ_p for a functor T is expressible in the Moss' language for T if there exists a natural translator τ for λ_p that is logically representable.*

Using this construction in the particular case of Kripke polynomial functors we proved the following theorem.

Theorem. *Given a Kripke polynomial functor K, if we allow $K2$ conjunctions in Moss' language and the coalgebraic language, then every formula of monadic predicate liftings for K can be translated in toa formula of Moss' language for K. If K is finitary this translation is a finitary translation.*

During the proof of this last result we had to prove the following interesting result.

Theorem. *Let K_1 and K_2 be two functors such that:*

- *We can define Moss' language for both of them.*
- *Every singleton lifting for K_1 can be translated into the Moss' language of K_1 using logical translators.*
- *Every singleton lifting for K_2 can be translated into the Moss' language of K_2 using logical translators.*

If these three condition hold then

- *Every singleton lifting for $K_1 + K_2$ can be translated into the Moss' language for $K_1 + K_2$ using logical translators*
- *Every singleton lifting for $K_1 \times K_2$ can be translated into the Moss' language for $K_1 \times K_2$ using logical translators*
- *Every singleton lifting for $\mathcal{P}K_1$ can be translated into the Moss' language for $\mathcal{P}K_1$ using logical translators*

In other words logical translators can be extended over products, coproducts and composition with the functor $\mathcal{P}$. This result has the following immediate extension.

Let H be a functor which we can define Moss' language. We define the collection of *Kripke polynomial functors modulo H* inductively as follows:

$$K := \mathcal{I} \,|\, D \,|\, H \,|\, K_0 + K_1 \,|\, K_0 \times K_1 \,|\, Hom(D, K) \,|\, \mathcal{P}K.$$

In other words we add H as a base functor. Using the previous result we can then prove.

Theorem. *Let K be a Kripke polynomial functor modulo H. If every singleton lifting for H can be translated using logical translators, then every singleton lifting for K can be translated using logical translators.*

If we allow $K2$ conjunctions in Moss' language for K, then every predicate lifting for K can be translated into Moss' language for K.

What we have not done

The previous section summarized most of the result developed in this thesis. But there is much more work that is still to be done.

A first issue that we didn't solve is: Can we translate Moss' modality into languages of predicate liftings? We can define natural translators for Moss' modality as follows.

Definition. *A natural translator for nabla is a pair (ϑ, ρ) of natural transformations such that the following diagram*

$$\begin{tikzcd}
T\check{\mathcal{P}}A \arrow[r, "\vartheta_A"] \arrow[dr, "\nabla_A"'] & \prod_{i\in\eta} L_i(\check{\mathcal{P}})A \arrow[d, "\rho_A"] \\
& \check{\mathcal{P}}TA
\end{tikzcd} \tag{7.1}$$

commutes for every set A, where the L_i's are subfunctors of $\mathcal{P} + \Lambda + I$.

We can produce natural translators for all the Moss' modalities for Kripke polynomial functors. But we could not clarify under what conditions we can extract a syntactic translation from the pair (ϑ, ρ). The main problem is than now we want to obtain a one free variable formula from ρ. It is not yet clear for us what are the conditions for that extractions. The main problem is the case for $\mathcal{P}K$ functors.

Another issue related to the previous one is the translations of polyadic predicate liftings. We conjecture that such translations can be obtained using logical translators.

Related to the last theorem in the previous section it is out of the scope of our knowledge what base functors H allow us to translate singleton predicate liftings using logical translators.

Another issue is to investigate is about other constructions of functors that allow us to extend logical translators. Two interesting constructions to investigate are pullbacks and exponentials.

Bibliography

[1] Jiri Adámek, *Free algebras and automata realizations in the language of categories*, Commentationes Mathematicae Universitatis Carolinae **15** (1974), pp. 589–602.

[2] P Blackburn, M. de Rijke, and Y. Venema, *Modal logic*, Cambridge Tracts in Theoretical Computer Science, vol. 53, Cambridge University Press, 2002.

[3] Stanley N. Burris and H.P. Sankappanavar, *A course in universal algebra*, Graduate Texts in Mathematics, Springer, 1981.

[4] Robert Goldblatt, *Topoi, the categorical analysis of logic*, Studies in logic and the foundation of mathematics, vol. 98, North-Holland publishing company, 1979.

[5] Robert Goldblatt, *Final coalgebras and the hennessy-milner property.*, Ann. Pure Appl. Logic **138** (2006), no. 1-3, 77–93.

[6] Peter Gumm, *Elements of the general theory of coalgebras*, Tech. report, Rand Africaans University, Johannesburg, South Africa, 1999, Preliminary version.

[7] Bart Jacobs and Jan Rutten, *A tutorial on (co)algebras and (co)induction*, EATCS Bulletin **62** (1997), pp. 222–259.

[8] Lawrence S. Moss, *Coalgebraic logic.*, Ann. Pure Appl. Logic **96** (1999), no. 1-3, 277–317.

[9] Dirk Pattinson, *An introduction to the theory of coalgebras*, Course notes for NASSLLI, 2003.

[10] Lutz Schröder, *Expressivity of coalgebraic modal logic: The limits and beyond.*, FoSSaCS, 2005, pp. 440–454.

[11] Jaap van Oosten, *Basic category theory*, Aarhus lecture notes on category theory, 2002.

[12] Yde Venema, *Algebras and coalgebras*, P. Blackburn et al. ed., in Handbook of Modal Logic, ch. 6, pp. 331–426, Elsevier B.V, 2007.

Index