Master Thesis
Computing Science

Radboud University

A Coalgebraic Characterization of uioco

Author:
Menno Bartels
S1007797

First supervisor:
dr. J.C. Rot

Co-supervisor:
Ruben Turkenburg M.Sc.

Second reader:
dr. ir. G.J. Tretmans

July 22, 2024

Abstract

When developing software it is of great importance to be able to efficiently test the systems that are created. In the field of model based testing, the theory of uioco testing provides a framework for automatically generating and executing tests. In this thesis we work out a way to capture the uioco relation using the theory of coalgebra. This is achieved by considering constructions on systems, and then looking at the trace set produced by the result. We use distributive laws for defining these constructions for labeled transition systems and similar structures. In the end this approach produces the uioco characterization as a set of traces, satisfying certain properties. We capture these properties using a final coalgebra, and show that the constructions we described lead to systems that satisfy these properties.

Contents

Chapter 1

Introduction

In the development of software it is of great importance to be able to efficiently test the systems that are created. When checking a software system, one has to decide whether the system meets the requirements that were set when designing it. This decision is often made by considering the input that a user can give to the system and the outputs that the system gives back as a response.

The field of model based testing [2, 15] tries to automate the process of testing software systems by developing theory and tools that autonomously generate and run tests from models. A popular way to model systems is by using labeled transition systems. These are a type of transition structure with states, where we move between states using so called labels. A label can be an input, which models that a user manipulates the system, or an output, which models behavior that the system may show to a user.

In model based testing we compare between two systems, the specification and the implementation. Both of these systems are often modeled as labeled transition systems. We see the specification as a description of the behavior we want a system to have, and the implementation as an attempt to match the wanted behavior. Having a specification and an implementation, we need to decide if the implementation indeed plays by the rules that the specification lays out. This is can be determined using a conformance relation.

One of these conformance relations is the universal input output conformance (uioco) relation [20]. This relation compares the systems by using sequences of labels that are specified in the specification. These sequences of labels are called traces. For both the implementation and specification we look at the outputs that the systems provides after performing a trace. The uioco relation then states that the implementation conforms to the specification if the implementation does not show any output that can not be matched in the specification. In other words, the implementation behaves correctly, if for all the cases, we do not observe unexpected output.

In model based testing theory, labeled transition systems are often described as tuples containing a set of states, a set of input and output labels, a set of transitions and a initial state. An alternative way to define a labeled transition system is by using coalgebras [12]. The field of coalgebra is a well established field of study, which is based on category theory [5]. Using coalgebra we can show that labeled transition systems can also be described by providing the set of states and a transition map of a specific type. This different way of approaching labeled transition systems also gives a different way to approach the uioco relation. Coalgebra lets us describe the behavior of a system by considering final coalgebras, which form a central notion in coalgebra theory. In the case of deterministic labeled transition systems this will give that the behavior of a system is precisely described by the set of traces that it can produce. In recent work [11] we have seen that the predecessor of the uioco relation, ioco, was used as an example of a coalgebraic uncertain bisimulation. Due to this result we set out to get a better understanding of these relations coalgebraically.

This brings us to the main goal of this thesis. We want to use the theory of coalgebra to give a coalgebraic formalization of constructions found in uioco theory. With this we aim to bridge the gap between the fields of model based testing and coalgebra theory. We do this by following the work of Janssen [6], in which we find several approaches to describe the ioco and uioco relations. These involve the usual definition, a syntactical approach using suspension languages, and an approach using constructions on labeled transition systems.

We focus on the method involving the constructions, for only the uioco relation. This approach is the best suited candidate for a coalgebraic translation, as the constructions involved can be encoded in coalgebra quite naturally. The approach for uioco specifically is also less complicated than the approach for ioco, as it involves less steps. A last motivation for choosing uioco instead of ioco is that it is also argued to be the preferred relation when it comes to conformance testing [19]. In this thesis we show that the constructions can be described as functors between categories of suited types of coalgebras. We use final coalgebras to show that the systems we obtain after the constructions adhere to the expected properties, according to [6].

In chapter 2 we recall definitions we need to define the uioco conformance relation. In chapter 3 we write down the prerequisites we need from coalgebra to be able to work out the constructions. In chapter 4 we discuss how we can see LTSes as coalgebras, and give the coalgebraic versions of the LTS constructions. In chapter 5 we show that considering the traces of a coalgebra resembled its behavior, by providing the final coalgebra. Lastly, in chapter 6 we show that the constructions yield coalgebras satisfying certain properties by considering the final coalgebra for systems that we obtain after the constructions.

Chapter 2

Preliminaries and Problem Statement

In this thesis, we will use labeled transition systems (LTS) as models and the uioco relation [6, 16, 17] as the conformance relation. This is one of the approaches for conformance testing we find in the field of model based testing. A conformance relations tells us when the observed behavior of a system is deemed to be correct. For the uioco relation this means that two systems should agree on their output after performing a sequence of labels. Such a sequence of labels is called a trace.

The uioco relation can also be described using a so-called trace characterization. Such a characterization gives that two systems are also related by uioco if the trace characterization of one system is a subset of the trace characterization of the other system. We can obtain such a characterization from the definition of uioco, but also via a series of LTS transformations.

In this chapter we give an overview of the relevant definitions and constructions. The main goal of this chapter is to give a sketch of the construction and to serve as an introduction to the coalgebraic formalization that follows in the later sections. For a more detailed version of the construction we refer to [6].

2.1 Labeled Transition Systems

Labeled transition systems (LTS) [12, 15] are often used to model real world systems. In our case we use LTSes with inputs and outputs. Such LTSes can model the behavior of a real world system, in terms of system state, inputs and outputs. They are defined as follows.

Definition 2.1. A labeled transition system (LTS) with inputs and outputs is a 5-tuple $\mathcal{S} = (Q, I, O, T, q_0)$ where Q is the set of states, $\mathcal{I}$ is the set of input labels, O is the set of output labels, $T \subseteq Q \times (I \cup O) \times Q$ is the transition relation and $q_0 \in Q$ is the initial state.

Remark 2.2. In this thesis we do not consider silent steps for labeled transition systems [15]. This is because silent steps do not behave as regular labels, and should thus be handled with some care. For example, when considering traces of systems, we would have to make sure that the label used for silent steps does not end up in the traces produced by a system. This would then mean that when giving definitions for labeled transition systems with silent steps, we have to take into account such requirements. We do however not see any problems in extending the theory in this thesis so that they do involve these silent steps.

When it is clear from context that we are considering inputs and outputs we also often just write "LTS" instead of "LTS with inputs and outputs". We extend the single step transition relation to a relation that involves traces of labels. For the disjoint union of input and output labels we write $\Sigma = I + O$.

Definition 2.3. Let $\mathcal{S} = (Q, I, O, T, q_0)$ be an LTS. Let $q, q' \in Q$, $a \in \Sigma$ and $\sigma \in \Sigma^*$. We inductively define the relation $\rightarrow \subseteq Q \times \Sigma^* \times Q$ with the rules:

$$\frac{}{q \xrightarrow{\varepsilon} q}\ (1) \qquad\qquad \frac{\exists q'' \in Q \colon (q, a, q'') \in T \quad q'' \xrightarrow{\sigma} q'}{q \xrightarrow{a\sigma} q'}\ (2)$$

From the above rules, we can see that for an LTS $\mathcal{S} = (Q, I, O, T, q_0)$, we may write $q \xrightarrow{a} q'$, if $(q, a, q') \in T$. In a similar way we may write $q \overset{a}{\not\rightarrow} q'$, if $(q, a, q') \notin T$. If for all $q' \in Q$ we have that $q \overset{a}{\not\rightarrow} q'$ we omit the resulting state and simply write $q \overset{a}{\not\rightarrow}$.

If for a state $q \in Q$ and a label $a \in \Sigma$ there exists a state $q' \in Q$ such that $q \xrightarrow{a} q'$ we call the label a *enabled* for q. We say that an LTS is *input enabled* if for all states, all input labels are enabled. We say an LTS is *non-blocking* if for every state there is at least one enabled output label. We denote the collection of LTSes as $\mathcal{LTS}$, and the collection of input enabled LTSes as $\mathcal{LTS}_{\mathrm{IE}}$. An LTS is *deterministic* if for all $q \in Q$ and $a \in \Sigma$ there is at most one $q' \in Q$ such that $q \xrightarrow{a} q'$. We often draw LTSes as diagrams, as we can see from the following example. Here we use a question mark and an exclamation point to distinguish between input and output labels respectively.

Example 2.4. Let $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $I = \{\mathit{button}\}$ and $O = \{\mathit{coffee}, \mathit{tea}\}$. Define $T = \{(q_0, \mathit{button}, q_1), (q_0, \mathit{button}, q_3), (q_1, \mathit{coffee}, q_2), (q_3, \mathit{tea}, q_4)\}$. Now $\mathcal{S} = (Q, I, O, T, q_0)$ is an LTS with input and output. We can depict $\mathcal{S}$ as the following diagram.

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "?\mathit{button}"] \arrow[dr, "?\mathit{button}"'] & q_1 \arrow[r, "!\mathit{coffee}"] & q_2 \\
& & q_3 \arrow[r, "!\mathit{tea}"] & q_4
\end{tikzcd}$$

From the inductively defined transition relation "$\rightarrow$", we can also derive the set of traces that a state produces.

Definition 2.5. Let $\mathcal{S} = (Q, I, O, T, q_0)$ be an LTS with inputs and outputs and let $\sigma \in \Sigma^*$. For a state $q \in Q$ we define the trace set of q as

$$\sigma \in \mathsf{traces}(q) \overset{\text{def}}{\iff} \exists q' \in Q \colon q \xrightarrow{\sigma} q'$$

Example 2.6. Let $\mathcal{S}$ be the LTS from example 2.4. Then the trace set for the initial state is $\mathsf{traces}(q_0) = \{\varepsilon, \textit{button}, \textit{button coffee}, \textit{button tea}\}$.

2.1.1 Quiescence

Quiescence is a special kind of output, which is used to denote the absence of regular outputs. A state that has no enabled regular output labels is called a *quiescent* state. If we think of LTSes with inputs and outputs as real world systems, quiescence means that the system does not give any output, and waits for a user to give input.

Definition 2.7. Let $\mathcal{S} = (Q, I, O, T, q_0)$ be an LTS with inputs and outputs. A state $q \in Q$ is quiescent if for all $o \in O$ it holds that $q \overset{o}{\not\rightarrow}$.

Now that we know what quiescence is, we can define Quiescent Labeled Transition Systems (QLTS) [15]. In a QLTS we have a way to explicitly consider quiescent states. We add a special output label δ which models quiescent behavior. If in a QLTS the label δ is enabled for some state, it should hold that the state that is reached after taking the δ-transition is quiescent. It is important to note that it can happen that a regular output label o and the label δ are both enabled in a state. This models the situation where the output o may or may not occur, which can happen in a non-deterministic setting. For the disjoint union of the set of outputs and the label δ we write $O^\delta = O + \{\delta\}$, and we write $\Sigma^\delta = I + O^\delta$ for the set of all labels including δ.

Definition 2.8. A quiescent labeled transition system is an LTS $\mathcal{S} = (Q, I, O^\delta, T, q_0)$ where for transitions $q \xrightarrow{\delta} q'$ it holds that q' is quiescent and that $q' \xrightarrow{\delta} q'$.

2.1.2 Deltafication

As we have seen from the definitions above, we can observe that a state is quiescent by looking at the internal structure of the LTS. To be able to explicitly observe quiescence we define a construction called deltafication, which adds δ self-loops to quiescent states. This then creates a QLTS out of an LTS.

Definition 2.9. Let $\mathcal{S} = (Q, I, O, T, q_0)$ be an LTS. We define the deltafication of $\mathcal{S}$ as the QLTS

$$\Delta(\mathcal{S}) = (Q, I, O^\delta, T', q_0)$$

where

$$T' = T \cup \{(q, \delta, q) \mid q \in Q, q \text{ is quiescent}\}$$

Lemma 2.10. Let $\mathcal{S} = (Q, I, O, T, q_0)$ be an LTS. Then $\Delta(\mathcal{S}) = (Q, I, O^\delta, T', q_0)$ is a QLTS.

Proof. The only δ-transitions in $\Delta(\mathcal{S})$ are loops on quiescent states in $\mathcal{S}$. This means that for any transitions of the form $q \xrightarrow{\delta} q'$ for $q, q' \in Q$ it holds that $q = q'$, hence q' is also quiescent and there is a transition $q' \xrightarrow{\delta} q'$. □

Example 2.11. Recall the LTS $\mathcal{S}$ from example 2.4. If we now apply deltafication on $\mathcal{S}$, we get the QLTS $\Delta(\mathcal{S})$, which has added δ self-loops on states q_0, q_2 and q_4, as these states are quiescent.

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[loop above, "\delta"] \arrow[r, "?\mathit{button}"] \arrow[dr, "?\mathit{button}"'] & q_1 \arrow[r, "!\mathit{coffee}"] & q_2 \arrow[loop right, "\delta"] \\
& & q_3 \arrow[r, "!\mathit{tea}"] & q_4 \arrow[loop right, "\delta"]
\end{tikzcd}$$

2.2 The uioco relation

Using quiescence and deltafication, we can define the uioco relation. This relation is often used as a conformance relation in the field of model based testing [19, 6], and is the successor of the ioco relation [17, 16]. A conformance relation is used to define when an implementation is deemed correct according to a specification, i.e. the implementation conforms to the specification. To explain what this means for the uioco relation, we first need to specify a special kind of traces that the uioco relation considers. These traces are so called universal traces [20]. A trace $\sigma \in (\Sigma^\delta)^*$ is called a universal trace if, for all $\sigma_1, \sigma_2 \in (\Sigma^\delta)^*$ and $i \in I$ such that we can write $\sigma = \sigma_1 i \sigma_2$, it hold that i is enabled in the states that σ_1 leads to. Note that in a deterministic system, the universal trace set and the trace set are equal. This is because any σ_1 always leads to only one state. Then if the input i was not enabled, σ would not be a trace of the LTS to begin with. Hence, any trace is then a universal trace.

Definition 2.12. Let $\mathcal{S} = (Q, I, O, T, q_0)$ be an LTS. We define the universal traces of $\mathcal{S}$ as

$$\begin{aligned}\mathsf{Utraces}(\mathcal{S}) \;=\; & \{\sigma \in \mathsf{traces}(\Delta(\mathcal{S})) \mid \forall \sigma_1, \sigma_2 \in (\Sigma^\delta)^*, \forall i \in I\colon \\ & \sigma = \sigma_1 i \sigma_2 \implies (\forall q \in Q\colon (q_0 \xrightarrow{\sigma_1} q \implies (\exists q' \in Q\colon q \xrightarrow{i} q')))\}\end{aligned}$$

Example 2.13. Consider the following deltafied LTS $\Delta(\mathcal{S})$.

$$\begin{tikzcd}
{} \arrow[d] & \\
p_0 \arrow[loop left, "\delta"] \arrow[d, "?\mathit{button}"'] \arrow[dr, "?\mathit{button}"] & \\
p_1 \arrow[d, "!\mathit{tea}"'] & p_3 \arrow[loop right, "\delta"] \arrow[d, "?\mathit{button}"] \\
p_2 \arrow[loop left, "\delta"] & p_4 \arrow[d, "!\mathit{coffee}"] \\
 & p_5 \arrow[loop left, "\delta"]
\end{tikzcd}$$

We give an example of a trace that is universal, and a trace that is not universal:

- $\mathit{button}\ \delta\ \mathit{button} \in \mathsf{Utraces}(p_0)$. If we take $\sigma_1 = \varepsilon$, $\sigma_2 = \delta\ \mathit{button}$ we get that $p_0 \xrightarrow{\varepsilon} p_0$, and *button* is enabled for p_0. If we take $\sigma_1 = \mathit{button}\ \delta$ and $\sigma_2 = \varepsilon$, we get that σ_1 only leads to one state, namely $p_0 \xrightarrow{\mathit{button}\ \delta} p_3$, and *button* is enabled in p_3. These are all the possible decompositions of $\mathit{button}\ \delta\ \mathit{button}$.
- $\mathit{button}\ \mathit{button} \notin \mathsf{Utraces}(p_0)$. We take $\sigma_1 = \mathit{button}$ and $\sigma_2 = \varepsilon$. Then we have that $p_0 \xrightarrow{\mathit{button}} p_1$ and $p_0 \xrightarrow{\mathit{button}} p_3$. In state p_1 the input label *button* is not enabled.

We can now explain what it means for an implementation to conform to a specification according to uioco. We first remark that we assume that implementations are input enabled. An implementation resembles a model of a real world system that we are testing. For such systems we assume the testability assumption [3]. This claims that every real world system can be seen as a black box, where all the inputs are enabled. To put it differently, the testability assumption states that we can always press all of the buttons that are on our system.

Given an implementation $\mathcal{I} \in \mathcal{LTS}_{\mathrm{IE}}$ and a specification $\mathcal{S} \in \mathcal{LTS}$, the uioco relation considers the universal traces of the specification $\mathcal{S}$. This is because the specification is the LTS that describes the behavior that is deemed correct. Moreover, for the traces that are not part of the universal traces of $\mathcal{S}$, the implementation is free to have any behavior we want. Given a universal trace $\sigma \in \mathsf{Utraces}(\mathcal{S})$ the uioco relation requires $\mathcal{I}$ to match all of the output labels that $\mathcal{S}$ has enabled after σ. If this is the case for all universal traces of $\mathcal{S}$, we say that $\mathcal{I}$ is uioco-conformal to $\mathcal{S}$.

Definition 2.14. Let $\mathcal{S} = (Q^{\mathcal{S}}, I, O, T^{\mathcal{S}}, q_0^{\mathcal{S}})$ be an LTS. Let $\mathcal{I} = (Q^{\mathcal{I}}, I, O, T^{\mathcal{I}}, q_0^{\mathcal{I}})$ be an input enabled LTS. We define the relation $\mathsf{uioco} \subseteq \mathcal{LTS}_{\text{IE}} \times \mathcal{LTS}$ as

$$\mathcal{I} \;\mathsf{uioco}\; \mathcal{S} \overset{\text{def}}{\iff} \forall \sigma \in \mathsf{Utraces}(\mathcal{S}), \forall o \in O^{\delta}:$$
$$(\exists q \in Q^{\Delta(\mathcal{I})} \colon q_0^{\Delta(\mathcal{I})} \xrightarrow{\sigma o} q) \implies (\exists q \in Q^{\Delta(\mathcal{S})} \colon q_0^{\Delta(\mathcal{S})} \xrightarrow{\sigma o} q)$$

Example 2.15 ([18])**.** Let $I = \{\textit{button}\}$ and let $O = \{\textit{coffee}, \textit{tea}\}$. Consider the input enabled QLTS $\Delta(\mathcal{I})$ on the left, and the QLTS $\Delta(\mathcal{S})$ on the right.

$$\begin{tikzcd}
{} \arrow[d] \\
q_0 \arrow[loop left, "\delta"] \arrow[d, "?button"] \\
q_1 \arrow[loop left, "\delta"] \arrow[d, "?button"] \\
q_2 \arrow[loop left, "?button"] \arrow[d, "!tea"] \\
q_3 \arrow[loop left, "{?button,\delta}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
{} \arrow[d] & \\
p_0 \arrow[loop left, "\delta"] \arrow[d, "?button"'] \arrow[dr, "?button"] & \\
p_1 \arrow[d, "!tea"'] & p_3 \arrow[loop right, "\delta"] \arrow[d, "?button"] \\
p_2 \arrow[loop left, "\delta"] & p_4 \arrow[d, "!coffee"] \\
& p_5 \arrow[loop left, "\delta"]
\end{tikzcd}$$

We have that $\mathcal{I} \neg\mathsf{uioco}\; \mathcal{S}$, as for $?button\ \delta\ ?button \in \mathsf{Utraces}(\Delta(\mathcal{S}))$ we have that $q_2 \xrightarrow{tea} q_3$, but $p_4 \overset{tea}{\not\rightarrow}$. In order to make $\mathcal{I}$ an implementation that does conform to $\mathcal{S}$, we add a output transition from q_1 to get the input enabled LTS $\mathcal{I}'$. Notice that if we now look at $\Delta(\mathcal{I}')$ below, we see that the δ loop on q_1 is no longer there. This is because making the output label *tea* enabled for q_1, makes it so that q_1 is no longer quiescent. The LTS $\Delta(\mathcal{I}')$ now looks as follows:

$$\begin{tikzcd}
{} \arrow[d] & \\
q_0 \arrow[loop left, "\delta"] \arrow[d, "?button"] & \\
q_1 \arrow[d, "?button"'] \arrow[dr, "!tea"] & \\
q_2 \arrow[loop left, "?button"] \arrow[d, "!tea"] & q_4 \arrow[loop right, "{?button,\ \delta}"] \\
q_3 \arrow[loop left, "{?button,\ \delta}"] &
\end{tikzcd}$$

For $\mathcal{I}'$ we see that for $\sigma = ?button\ \delta\ ?button$ it holds that $q_0 \overset{\sigma}{\not\rightarrow}$, hence the claim in the uioco relation holds vacuously for this trace. Moreover we have that $\mathcal{I}'\ \mathsf{uioco}\ \mathcal{S}$.

Remark 2.16. Note that $\mathcal{LTS}_{\text{IE}} \subseteq \mathcal{LTS}$, hence the uioco relation is also defined for two input enabled LTSes.

2.2.1 The uioco Characterization

The given definition of the uioco relation defines when the behavior of an implementation is correct according to some specification. We can also give such a definition of correctness for single traces. We say that a trace conforms to a specification, if it occurs in the trace set of a deltafied uioco-conforming implementation. The reason such an implementation has to be deltafied, is because the traces may include the label δ, meaning that we consider quiescence explicitly. Deltafication ensures that for the implementations quiescence is considered explicitly. A trace that can occur in such a uioco conforming implementation is called a *uioco conformal* trace. If we take the set of all such traces, we get to the definition of the uioco characterization.

Definition 2.17. Let $\mathcal{S} = (Q, I, O, T, q_0) \in \mathcal{LTS}$. Let $\sigma \in (\Sigma^{\delta})^*$. We define

$$\sigma \textsf{ uiocfl } \mathcal{S} \overset{\text{def}}{\iff} \exists \mathcal{I} \in \mathcal{LTS}_{\text{IE}} \colon \mathcal{I} \textsf{ uioco } \mathcal{S} \wedge \sigma \in \textsf{traces}(\Delta(\mathcal{I}))$$

Then the uioco conformal trace characterization is defined as

$$\langle \mathcal{S} \rangle_{\textsf{uioco}} = \{\sigma \in (\Sigma^{\delta})^* \mid \sigma \textsf{ uiocfl } \mathcal{S}\}$$

Example 2.18. Recall the LTS $\mathcal{S}$ from example 2.4. For $\mathcal{S}$ we can see that every trace that is produced is also a universal trace. For these traces we know that they can occur in a uioco conforming implementation, as we have seen in for example 2.15. This gives that at least the traces *button*, *button coffee* and *button tea* are in the uioco characterization. The states q_0, q_2 and q_4 are quiescent. This gives that the uioco characterization should also include traces of the form δ^n, δ^n *button*, δ^n *button coffee* , δ^n *button tea* and δ^n *button tea* δ^n for $n \in \mathbb{N}$. Any underspecified trace is also in the uioco characterization. These are traces that start with δ^n *button button*, δ^n *button coffee* δ^n *button* or δ^n *button tea* δ^n *button* for $n \in N$.

2.3 LTS Constructions

In the above definition of the uioco characterization we use the definition of uioco conformal traces. This does not give us a direct way to actually obtain the uioco characterization from a given LTS. As we will see, we can also obtain the uioco characterization by applying transformations on an LTS, and then considering the trace sets of the resulting LTS. These constructions involve deltafication, which we have already defined, and chaotic completion and determinization, which we both define in this section.

2.3.1 Chaotic Completion

According to the uioco relation a trace is underspecified if at any point in the trace we arrive in a state for which the next input label in the trace is not enabled. For a deterministic system this means that all traces are specified, as each trace has only one path through the system. For non-deterministic systems it can happen that at some point in a trace we can end up in multiple states. This makes it possible for a non-deterministic system to have traces which are underspecified according to uioco.

Chaotic completion is used to model freedom of implementation for underspecified traces. When a trace is underspecified, the implementer may allow any behavior. This means that a conforming implementation can have any continuation of an underspecified trace in its trace set. We perform the operation of chaotic completion to explicitly add the underspecified traces to our system. This is achieved by adding transitions to a so-called chaotic state q_χ [6]. For a chaotic state it holds that any trace which can appear in the trace set of a possible implementation, also occurs in the trace set of the chaotic LTS. In other words, we want that for any $\mathcal{I} \in \mathcal{LTS}_{\text{IE}}$, we have that for all $\sigma \in \mathsf{traces}(\mathcal{I})$ we also have $\sigma \in \mathsf{traces}(q_\chi)$. There are many QLTSes having a chaotic state, but we use the following one, as it is minimal.

Definition 2.19. We define the chaotic QLTS $\chi = (\{q_\chi, q_\Omega\}, I, O^\delta, T_\chi, q_\chi)$ as the QLTS defined by the following diagram:

$$\begin{tikzcd}
{} \arrow[r] & q_\chi \arrow[loop above, "I{,}O"] \arrow[r, bend left, "\delta"] & q_\Omega \arrow[loop above, "\delta"] \arrow[l, bend left, "I"]
\end{tikzcd}$$

We use this chaotic QLTS χ for defining the construction of chaotic completion.

Definition 2.20. Let $\mathcal{S} = (Q, I, O^\delta, T, q_0)$ be an QLTS. We define the chaotic completion of $\mathcal{S}$ as the QLTS

$$\Xi(\mathcal{S}) = (Q + \{q_\chi, q_\Omega\}, I, O^\delta, T', q_0)$$

where

$$T' = T \cup T_\chi \cup \{(q, a, q_\chi) \mid q \in Q, a \in I, \neg\exists q' \in Q \colon q \xrightarrow{a} q'\}$$

Example 2.21. Consider the deltafied LTS $\Delta(\mathcal{S})$ from example 2.11. If we apply chaotic completion to that LTS we get the LTS $\Xi(\Delta(\mathcal{S}))$. We abbreviate the labels "*button*", "*coffee*" and "*tea*" with "*b*", "*c*" and "*t*" respectively, to increase readability. The transitions that are dashed are added in the completion, together with the chaotic LTS χ.

$$\begin{tikzcd}
 & & q_1 \arrow[r, "!c"] \arrow[drr, dashed, "?b"] & q_2 \arrow[loop above, "\delta"] \arrow[dr, dashed, "?b"] & & \\
{} \arrow[r] & q_0 \arrow[loop above, "\delta"] \arrow[ur, "?b"] \arrow[dr, "?b"'] & & & q_\chi \arrow[loop above, "?b{,}!c{,}!t"] \arrow[r, bend left, "\delta"] & q_\Omega \arrow[loop above, "\delta"] \arrow[l, bend left, "?b"] \\
 & & q_3 \arrow[r, "!t"'] \arrow[urr, dashed, "?b"] & q_4 \arrow[loop below, "\delta"] \arrow[ur, dashed, "?b"] & &
\end{tikzcd}$$

2.3.2 Determinization

Once we have applied deltafication and chaotic completion on an LTS, the last construction we need is determinization. Determinization creates a deterministic LTS out of any LTS. This is achieved by using the powerset construction which is a well-known construction from automata theory [4, 14]. We know that using the powerset construction, determinization will preserves the trace set that is produced by a system. This would imply that we do not actually need it, as we are going to consider the trace set of the LTS we obtain after applying determinization. However, it turns out that for the coalgebraic notion of equivalence we are interested in, we can not look at the traces of non-deterministic systems. This is because the behavior of a non-deterministic system is not described by only the traces it can produce, but also by the branching structure of the system. The powerset construction gets rid of this, by creating an LTS where each trace has a single path through the system. This then makes it so that for such systems, the behavior of the system is correctly captured by only considering the traces.

The construction of determinization creates a deterministic LTS out of a non-deterministic one in the following way. The states of the deterministic LTS are defined as subsets of states from the original LTS. Then given a state of the deterministic LTS (which is a subset of states of the original LTS) and a label, we determine all of the states in the original LTS we can reach from any state in the subset, using the label. Then putting together all of these reachable states we again get a subset of the states of the original LTS. We then define that set as the result of taking the transition using the label in the new deterministic LTS.

Definition 2.22. Let $\mathcal{S} = (Q, I, O^\delta, T, q_0)$ be a QLTS. We define the determinization of $\mathcal{S}$ as the LTS

$$\mathsf{det}(\mathcal{S}) = (\mathcal{P}(Q), I, O^\delta, T', \{q_0\})$$

where

$$T' = \{(t, a, t') \mid t, t' \in \mathcal{P}(Q), a \in \Sigma^\delta, \forall q' \in t', \exists q \in t \colon (q, a, q') \in T\}$$

Example 2.23. We consider the LTS $\Xi(\Delta(\mathcal{S})))$ from example 2.21. If we apply determinization to that LTS we get the LTS $\mathsf{det}(\Xi(\Delta(\mathcal{S})))$.

$$\begin{tikzcd}
 & & & \{q_2\} \arrow[loop above, "\delta"] \arrow[dr, "?b"] & & \\
{} \arrow[r] & \{q_0\} \arrow[loop above, "\delta"] \arrow[r, "?b"] & \{q_1, q_3\} \arrow[ur, "!c"] \arrow[rr, "?b"] \arrow[dr, "!t"'] & & \{q_\chi\} \arrow[loop above, "{?b,!c,!t}"] \arrow[r, bend left, "\delta"] & \{q_\Omega\} \arrow[loop above, "\delta"] \arrow[l, bend left, "?b"] \\
 & & & \{q_4\} \arrow[loop below, "\delta"] \arrow[ur, "?b"'] & &
\end{tikzcd}$$

2.3.3 The uioco Characterization by Construction

Using the definitions we have now seen, we get a constructive way to capture the uioco characterization. This is achieved by taking any LTS, applying deltafication, chaotic completion and determinization, and then considering the trace set.

Theorem 2.24. Let $\mathcal{S} \in \mathcal{LTS}$. Then

$$\langle \mathcal{S} \rangle_{\mathsf{uioco}} = \mathsf{traces}(\mathsf{det}(\Xi(\Delta(\mathcal{S}))))$$

Proof. We refer to [6]. $\square$

In the rest of this thesis we show that deltafication, chaotic completion and determinization can all be described as maps between collections of LTSes. Then for the LTSes we obtain after these constructions, we show that the map traces maps each such LTS to its behavior. In our case, this behavior is resembled by the set of traces that the LTS can produce.

2.4 Trace set properties

In Janssen's work [6] we find a third way to describe the uioco characterizations. This is done by using so called *suspension languages*. We will not go into detail on the constructions that Janssen defines for trace sets to obtain the characterization via suspension languages, but we do use the properties that define suspension languages [21]. As it turns out, we can show that these properties are ensured by the constructions we formalize. In [6] it is already shown that the non-coalgebraic construction also has these properties. However, in order to show that our formalization yields similar results, we also translate these properties to coalgebra, and show that our formalization adheres to them. This aids towards a correctness claim for the formalization to come.

Definition 2.25. Let $\Sigma^\delta = I + O^\delta$ be a set of labels. A set $L \subseteq (\Sigma^\delta)^*$ is

Prefix closed	$\Leftrightarrow$	$\forall \sigma \in (\Sigma^\delta)^*, \forall l \in \Sigma^\delta \colon \sigma l \in L \implies \sigma \in L$
Non-blocking	$\Leftrightarrow$	$\forall \sigma \in L,\ \exists o \in O^\delta \colon \sigma o \in L$
Input enabled	$\Leftrightarrow$	$\forall \sigma \in L,\ \forall i \in I \colon \sigma i \in L$
Anomaly free	$\Leftrightarrow$	$\forall \sigma \in L,\ \neg\exists \sigma_1, \sigma_2 \in (\Sigma^\delta)^*,\ \neg\exists o \in O \colon \sigma = \sigma_1 \delta o \sigma_2$
Quiescence reducible	$\Leftrightarrow$	$\forall \sigma, \rho \in (\Sigma^\delta)^* \colon \sigma\delta\rho \in L \implies \sigma\rho \in L$
Quiescence stable	$\Leftrightarrow$	$\forall \sigma, \rho \in (\Sigma^\delta)^* \colon \sigma\delta\rho \in L \implies \sigma\delta\delta\rho \in L$

If $L \subseteq (\Sigma^\delta)^*$ is non-empty and satisfies all of the above properties, L is called a suspension language. We denote the set of all suspension languages as $\mathcal{SL}$.

Theorem 2.26. Let $\mathcal{S} \in \mathcal{LTS}$. The uioco characterization $\langle \mathcal{S} \rangle_{\mathsf{uioco}}$ is a suspension language.

Proof. We refer to [6]. $\square$

Remark 2.27. In [6] we find that for the conformance relation of ioco we have that for any suspension language L, it holds that there exists an input enabled LTS $\mathcal{I}$ such that

$$\langle \mathcal{I} \rangle_{\mathsf{ioco}} = L$$

This is showed by considering constructions defined on trace sets. However, this takes into account underspecified traces according to the ioco relation instead of the uioco relation. As underspecified traces are not the same for both relations, this fact does not immediately transfer over from the ioco relation. This means that a similar theorem does not immediately hold for uioco. In [6] it is only stated that it is not possible, but it is not considered in detail. In the section 6.2 we do get a somewhat similar result, which states that for any suspension language we get an LTS that has exactly that suspension language as its trace set. This however, is not exactly the same as the theorem above. Investigating if such a theorem also holds for the uioco characterization is left as future work.

2.5 Main Goal and Contributions

We have now seen the uioco characterization, a construction that gives a way to algorithmically get the characterization, and properties that the characterization satisfies. Now the main question is: can we do this using coalgebra?

An LTS is often used as an example of a non-deterministic system for coalgebra [12, 5, 8]. With this in mind, we can show that the LTS constructions can be seen as functors between spaces of coalgebras. For example, deltafication will be a functor from the space of LTSes to the space of QLTSes. To show that our formalization also results in a suspension language, we need to do two things. First, we have to translate the suspension language properties for LTSes to the theory of coalgebra. Secondly, we need to show that the LTSes we obtain after the formalized constructions indeed produce trace sets that are suspension languages. It turns out that we can ensure the suspension language properties by ensuring similar properties on LTSes themselves. We show this by using the notion of final coalgebras, which are central in coalgebra theory.

With this formalization we try to bridge the gap between the fields of model based testing and coalgebra. On the one hand it is an attempt towards a better coalgebraic understanding of a concept in the field of model based testing. We have already seen that the ioco conformance relation was used as an example of the notion of a coalgebraic uncertain bisimulation [11]. This thesis tries to aid towards similar results for the uioco relation. On the other hand, our work showcases a construction, including types, which can be used by a tester to obtain the uioco characterization as a set of traces. Because we explicitly give the types of the intermediate LTSes we come across in the constructions, reproducing this construction would work especially well in a functional language. Also, the notion of final coalgebra could be used to create canonical uioco conforming implementation for a given specification. This last part is not worked out in this thesis, but is left as future work.

Chapter 3

Coalgebra Theory

In this chapter we recall the definitions from coalgebra that we need for formalizing the constructions mentioned in chapter 2. We give definitions of a coalgebra, a homomorphism between coalgebras, and categories of coalgebras. Then we recall a framework for defining functors between categories of coalgebras using natural transformations, based on theory from [10, 14, 5]. It turns out that the LTS constructions of deltafication, chaotic completion and determinization can all be described using this approach.

3.1 Coalgebras and Morphisms

To be able to give a coalgebraic formulation of the constructions discussed in [6] we first recall definitions for the theory we use. For the theory to come we assume primary knowledge in category theory. Because of this, some elementary category theory definitions are skipped, but can be found in for example [5].

Definition 3.1. Consider the category of sets and functions Set. For $X, Y \in \mathsf{Set}$ we write $X \times Y$ for products, which are implemented in Set by the Cartesian product, and $X + Y$ for coproducts, which are implemented by disjoint union. For the set $1 = \{\perp\}$, a map $f\colon X \to (Y + 1)$ is called a partial map, and we may also write $f\colon X \rightharpoonup Y$. If there exists $x \in X$ such that $f(x) \in Y$ we say that f is non-empty, which we denote by writing $f\colon X \rightharpoonup_{ne} Y$. If for $x \in X$, we have that $f(x) = \kappa_2(\perp)$ we say the result of $f(x)$ is undefined, and we may write just $f(x) = \perp$.

Definition 3.2. Given an endofunctor $F\colon \mathsf{Set} \to \mathsf{Set}$, an F-coalgebra is a tuple (X, c) where $X \in \mathsf{Set}$, and $c\colon X \to FX$ a function. An F-coalgebra homomorphism between two F-coalgebras (X, c) and (Y, d) is a map $h\colon X \to Y$ which makes the following diagram commute:

$$\begin{tikzcd} X \arrow[r, "h"] \arrow[d, "c"'] & Y \arrow[d, "d"] \\ FX \arrow[r, "Fh"] & FY \end{tikzcd}$$

The category where the objects are F-coalgebras, and the arrows are F-coalgebra homomorphisms, is denoted by $\mathsf{Coalg}(F)$. An F-coalgebra (Z, z) is final if for every F-coalgebra (X, c) there exists a unique F-coalgebra homomorphism $h\colon (X, c) \to (Z, z)$. For two elements $x, y \in X$ we say that they are behaviorally equivalent if for the unique F-coalgebra homomorphism $h\colon (X, c) \to (Z, z)$ it holds that $h(x) = (y)$. This notion is called behavioral equivalence, as final coalgebras are used to describe the behavior of coalgebras.

Example 3.3. Let $\Sigma \in \mathsf{Set}$ be a constant set, and let $X \in \mathsf{Set}$ be a set if states. Consider the functor $\mathcal{S}\colon \mathsf{Set} \to \mathsf{Set}$, partially defined by $\mathcal{S}(X) = \Sigma \times X$. An $\mathcal{S}$-coalgebra $(X, \langle o, t\rangle)$ is called a stream system over the set of labels Σ. Here $o\colon X \to \Sigma$ and $t\colon X \to X$ are functions. Considering a state $x \in X$, the function $o(x)$ gives the label that the state x produces and $t(x)$ gives the successor state of x. An F-homomorphism between F-coalgebras $(X, \langle o_X, t_X\rangle)$ and $(Y, \langle o_Y, t_Y\rangle)$ is a morphism $h\colon X \to Y$ that makes the following diagram commute:

$$\begin{tikzcd} X \arrow[r, "h"] \arrow[d, "\langle o_X, t_X\rangle"'] & Y \arrow[d, "\langle o_Y, t_Y\rangle"] \\ \Sigma \times X \arrow[r, "\mathrm{id}\times h"] & \Sigma \times Y \end{tikzcd}$$

In words this means that given a state $x \in X$, the outputs of x and $h(x)$ should match, and that first applying h and then considering the successor state, should give the same result as first looking at the successor state of x and then applying h to it. The final object in $\mathsf{Coalg}(\mathcal{S})$ is the coalgebra $(\Sigma^\omega, \langle i, d\rangle)$ where for $\sigma \in \Sigma^\omega$ we have $i(\sigma) = \sigma(0)$, which is the first label in σ, and $d(\sigma) = \sigma'$, which is σ without its first label, i.e. the derivative of σ. Here i denotes the first element of a trace, and d denotes the derivative of a trace. The unique homomorphism mapping an F-coalgebra (X, c) to $(\Sigma^\omega, \langle i, d\rangle)$ is the behavior map $beh\colon X \to \Sigma^\omega$, that sends each state to the stream it produces.

$$\begin{tikzcd} X \arrow[r, "beh"] \arrow[d, "\langle o, t\rangle"'] & \Sigma^\omega \arrow[d, "\langle i, d\rangle"] \\ \Sigma \times X \arrow[r, "\mathrm{id}\times beh"] & \Sigma \times \Sigma^\omega \end{tikzcd}$$

From this diagram we can derive that beh is uniquely defined as

$$beh(x)(0) = o(x) \quad beh(x)' = t(beh(x))$$

The next definition we give is that of a natural transformation. These give a way to transform one functor into another one while retaining the structure of the underlying category. In the theory to come, we use natural transformations to define the LTS constructions, in such a way that the result is still an LTS, i.e. a coalgebra.

Definition 3.4. Given two functors $F, G\colon \mathsf{Set} \to \mathsf{Set}$, a natural transformation is a collection of maps $\alpha_X : FX \to GX$ with $X \in \mathsf{Set}$, such that for every map $h\colon X \to Y$ the diagram below commutes. If α is a natural transformation we may write $\alpha\colon F \Rightarrow G$.

$$\begin{tikzcd}
FX \arrow[r, "\alpha_X"] \arrow[d, "Fh"'] & GX \arrow[d, "Gh"] \\
FY \arrow[r, "\alpha_Y"] & GY
\end{tikzcd}$$

3.2 Defining Functors via Natural Transformations

We now spell out a way to define functors between categories of coalgebras, via natural transformations. We do this by considering constructions from [14, 12, 5]. In later sections, the LTS constructions are then defined as such functors for categories of suited coalgebras.

For two functors $F, G : \mathsf{Set} \to \mathsf{Set}$ we often abbreviate the composition $F \circ G$ as just FG. We use this notation as it is a convention in coalgebra theory. When we compose functors with something which is not simply a functor, we do explicitly write the composition symbol. These conventions are for instance found in the following definition.

Definition 3.5. Let $F, G, H\colon \mathsf{Set} \to \mathsf{Set}$ be functors, and let $\alpha\colon HF \Rightarrow GH$ be a natural transformation. We define:

$$H_\alpha((X, c)) = (HX, \alpha_X \circ Hc)$$
$$H_\alpha(h) = Hh$$

We call $H_\alpha((X, c))$ the "H-α-lifting" of (X, c).

Proposition 3.6. The definition above has type $H_\alpha\colon \mathsf{Coalg}(F) \to \mathsf{Coalg}(G)$.

Proof. Take $(X, c) \in \mathsf{Coalg}(F)$. Then we have that $H_\alpha((X, c)) = (HX, \alpha_X \circ Hc)$. We know that $Hc\colon HX \to HFX$ and that $\alpha_{HX}\colon HFX \to GHX$, hence $\alpha_X \circ Hc\colon HX \to GHX$, hence $(HX, \alpha_X \circ Hc)$ is a G-coalgebra. □

Proposition 3.7. H_α is a functor.

Proof. We have already seen that H_α maps F-coalgebras to G-coalgebras. We now check that for an F-coalgebra homomorphism $h\colon X \to Y$, $H_\alpha(h) = Hh$ is a G-coalgebra homomorphism. Consider the following diagram:

$$\begin{tikzcd}
HX \arrow[r, "Hh"] \arrow[d, "Hc"'] & HY \arrow[d, "Hd"] \\
HFX \arrow[r, "HFh"] \arrow[d, "\alpha_X"'] & HFY \arrow[d, "\alpha_Y"] \\
GHX \arrow[r, "GHh"] & GHY
\end{tikzcd}$$

The top square commutes because h is an F-coalgebra homomorphism. The bottom square commutes because α is a natural transformation. Hence, we see that $H_\alpha(h) = Hh$ is a G-coalgebra homomorphism. Since H is a functor, we get that the identity and composition law for maps $H_\alpha(h) = Hh$ also hold. $\square$

Corollary 3.8. Let $F, G\colon \mathsf{Set} \to \mathsf{Set}$ be functors, and consider the identity functor $\mathrm{id}\colon \mathsf{Set} \to \mathsf{Set}$. Let $\alpha_X : F \Rightarrow G$ be a natural transformation. Then $\mathrm{id}_\alpha\colon \mathsf{Coalg}(F) \to \mathsf{Coalg}(G)$ is a functor.

We will see examples of applying this framework when we define the LTS constructions deltafication, chaotic completion and determinization.

Chapter 4

Deltafication, Chaotic Completion and Determinization as Functors

In Chapter 2 we have seen a standard definition of a labeled transition system and the constructions of deltafication, chaotic completion and determinization. In this chapter we translate these definitions to the theory of coalgebra. We point out that LTSes are actually coalgebras, and that the constructions on them can be defined as functors between categories of coalgebras.

4.1 Labeled Transition Systems as Coalgebras

Definition 4.1. Let $X \in \mathsf{Set}$ be a set of states and let $\Sigma \in \mathsf{Set}$ be a constant set of labels. We define the functor $F\colon \mathsf{Set} \to \mathsf{Set}$

$$F(X) = \Sigma \to \mathcal{P}(X)$$

Where for a function $f\colon X \to Y$ we define

$$\begin{aligned} F(f) :&\ F(X) \to F(Y) \\ F(f)(t)(a) =& \{f(y) \mid y \in t(a)\} \end{aligned}$$

Recall that earlier we defined an LTS as a tuple $\mathcal{S} = (Q, I, O, T, q_0)$. Consider the map $c\colon Q \to (\Sigma \to \mathcal{P}(Q))$ where for $q \in Q$ and $a \in \Sigma$ we have:

$$c(q)(a) = \{q' \in Q \mid (q, a, q') \in T\}$$

Now the tuple (Q, c) is an F-coalgebra which has the same transition behavior as the LTS defined as a tuple. Given a transition $t \in T$, the coalgebra has a matching transition encoded in the map c. There are some differences in notation however. The set of labels is now encoded in the functor, and we drop the initial state in the notation. Similar to what we had before, we can also draw F-coalgebras as transition diagrams.

Remark 4.2. In the field of coalgebra, a set of states is often denoted by the symbols X and Y instead of Q. When working out the similarity between an LTS and a coalgebra, we use the notation Q, but otherwise we will mostly use the notation we find in coalgebra theory.

Example 4.3. Let $X = \{x_0, x_1, x_2\}$ and $\Sigma = \{lock, write, free\}$. Then we define $c\colon X \to (\Sigma \to \mathcal{P}(X))$ as

$$\begin{array}{rcl} c(x_0)(lock) & = & \{x_1\} \\ c(x_1)(write) & = & \{x_2\} \\ c(x_2)(free) & = & \{x_0\} \end{array}$$

For the transitions that are not specified above, we get that all of them move to the empty set of states. For example:

$$c(x_0)(write) \quad = \quad \emptyset$$

Now the tuple (X, c) is an LTS. The diagram corresponding to this LTS the following:

$$\begin{tikzcd} x_0 \arrow[r, "lock"] & x_1 \arrow[r, "write"] & x_2 \arrow[ll, bend left, "free"] \end{tikzcd}$$

4.2 Labeled Transition Systems with Input and Output

In order to reason about uioco theory we need to consider LTSes with inputs and outputs. In the definition of an LTS as a we can achieve this by changing the functor for coalgebras we had in the previous section.

Definition 4.4. Let $X \in \mathsf{Set}$. Let I be a set of input labels and let O let be a set of output labels. We define the functor $F\colon \mathsf{Set} \to \mathsf{Set}$

$$F(X) = (I \to \mathcal{P}(X)) \times (O \to \mathcal{P}(X))$$

Let $\mathcal{S} = (Q, I, O, T, q_0)$ be an LTS. Let $i \in I$ and $o \in O$. We define the map $c\colon Q \to (I \to \mathcal{P}(Q)) \times (O \to \mathcal{P}(Q))$ in terms of its projections:

$$\begin{array}{rcl} \pi_1(c(q))(i) & = & \{q' \in Q \mid (q, i, q') \in T\} \\ \pi_2(c(q))(o) & = & \{q' \in Q \mid (q, o, q') \in T\} \end{array}$$

Now the tuple (Q, c) is an F-coalgebra. We fix the disjoint union of the input and output labels as $\Sigma = I + O$, set of all labels.

Remark 4.5. We could have also defined labeled transition systems with inputs and outputs using a functor of the form $F(X) = (I + O) \to \mathcal{P}(X)$. However, if we would do this, the definitions of the LTS constructions would become a bit more complicated. Using the functor in definition 4.4 above we can consider the input and output transitions separately using projection maps. This then gives a straightforward way to add transitions for input and output labels separately, which is what we want to be able to do when defining deltafication and chaotic completion.

Remark 4.6. Let $a \in \Sigma$ and consider $c\colon Q \to (I \to \mathcal{P}(Q)) \times (O \to \mathcal{P}(Q))$ as in definition 4.4 above. If it does not matter if the first or second projection is considered we abbreviate as follows:

$$c(q)(a) = \{q' \in X \mid (q, a, q') \in T\}$$

Notice that this notation is the same as we found in definition 4.1. However, when using this notation for labeled transition systems with input and output, it should be clear from context that implicitly we are considering the projections separately.

For LTSes as coalgebras we again give an inductively defined transition relation, which uses the notation we have seen before in definition 2.3.

Definition 4.7. Consider the functor $F(X) = (I \to \mathcal{P}(X)) \times (O \to \mathcal{P}(X))$. Let (X, c) be an F-coalgebra. Let $x, x' \in X$ and let $a \in \Sigma$. We inductively define the relation $\to \subseteq X \times \Sigma^* \times X$ with the rules:

$$\frac{}{x \xrightarrow{\varepsilon} x}\ (1) \qquad\qquad \frac{\exists x'' \in X\colon x'' \in c(x)(a) \quad x'' \xrightarrow{\sigma} x'}{x \xrightarrow{a\sigma} x'}\ (2)$$

In the theory to come, we also come across LTSes that have a slightly different base functor than the functor F that we have seen above. For these functors, it can happen that the definition of the transition relation "$\to$" does not exactly match the type of the functors. However, it should always be clear from context how we should alter the definition of "$\to$" to match the functor at hand.

We can also draw LTSes with inputs and outputs as diagrams. To distinguish between inputs and outputs in the diagram, we may write question marks and exclamation points respectively.

Example 4.8. Let $X = \{x_0, x_1, x_2, x_3, x_4\}$ and $I = \{\mathit{button}\}, O = \{\mathit{coffee}, \mathit{tea}\}$. The LTS (X, c) where

$$\begin{array}{rcl} c(x_0)(\mathit{button}) & = & \{x_1, x_3\} \\ c(x_1)(\mathit{coffee}) & = & \{x_2\} \\ c(x_3)(\mathit{tea}) & = & \{x_4\} \end{array}$$

For the transitions that are not specified above, we get that all of them move to the empty set of states. For example:

$$c(x_0)(\mathit{coffee}) \quad = \quad \emptyset$$

The diagram corresponding to this LTS the following:

$$\begin{tikzcd}
x_0 \arrow[r, "?button"] \arrow[dr, "?button"'] & x_1 \arrow[r, "!coffee"] & x_2 \\
 & x_3 \arrow[r, "!tea"] & x_4
\end{tikzcd}$$

4.3 LTS Constructions

Now we know that LTSes are actually coalgebras, we can formalize the constructions we have seen for them. We show that the constructions can be defined as functors between categories of coalgebras. We have seen that the constructions alter the structure of an LTS. In the theory of coalgebra this means that the functor which is used to define a coalgebra changes after we apply the construction. The reason for this is that the transition structure of an LTS is defined by the type of its functor.

Before we give the definitions of the LTS constructions coalgebraically we point out that the functors we are going to use are chosen in a specific way. Once we have defined all of the constructions, it will hold that we can concatenate them as in theorem 2.24 without the intermediate types mismatching. We could have also chosen to define the constructions in a more generic way so that they could be applied on more types of LTSes. We did not choose to do this as then we would have to resolve possible type mismatches when chaining the constructions together. If one would wish to apply these constructions on different types of LTSes, this can be achieved by slightly altering the definitions we give so that they match the types of the systems at hand.

We use the approach we have set up in section 3.2 to define the construction as functors.

4.3.1 Deltafication

The first construction we define in coalgebra is deltafication. For an F-coalgebra, deltafication adds the δ label as a loop on quiescent states. A state is quiescent if it has no output labels enabled. With adding these transitions, deltafication then changes the type of the base functor of the LTS such that quiescence is considered. For the rest of the theory we fix the functors F and F_Δ. For F_Δ the subscript Δ resembles the construction of deltafication.

$$\begin{array}{rclcl}
F(X) & = & (I \to \mathcal{P}(X)) & \times & (O \to \mathcal{P}(X)) \\
F_\Delta(X) & = & (I \to \mathcal{P}(X)) & \times & ((O + \{\delta\}) \to_{ne} \mathcal{P}(X))
\end{array}$$

We will see that deltafication transforms an F-coalgebra into an F_Δ-coalgebra. Recall that the subscript "*ne*" on the second projection means that there is at least one label in $O + \{\delta\}$ such that the result of the transition is defined. As the second projection maps to the powerset, we consider being undefined as being mapped to the empty set.

To be able to define deltafication, we first need to prepare some set up. This is because when taking a step in the LTS, i.e. applying the coalgebra structure, we lose information about the state we came from. We do need to keep track of this state to be able to define deltafication, because we want to be able to loop back to the state we came from. That is why we give the following constructions, where we save a copy of the state.

Definition 4.9. Let $G\colon \mathsf{Set} \to \mathsf{Set}$ be an arbitrary functor. Let (X, c) be a G-coalgebra. We define the functor $\mathrm{id}^{\times}\colon \mathsf{Coalg}(G) \to \mathsf{Coalg}(G \times \mathrm{id})$ as:

$$\begin{array}{rcl} \mathrm{id}^{\times}((X, c)) & = & (X, \langle c, \mathrm{id}\rangle) \\ \mathrm{id}^{\times}(h) & = & h \end{array}$$

If we take a state $x \in X$, and then write out $\langle c, \mathrm{id}\rangle(x) = (c(x), x)$, we see that indeed a copy of x is saved.

Definition 4.10. Let $X \in \mathsf{Set}$. Let $t \in F(X)$ and $x \in X$. Then let $o \in O$, $i \in I$ and let δ be the label representing quiescence. We define the components maps of natural transformation $\alpha\colon F \times \mathrm{id} \Rightarrow F_{\Delta}$, as follows:

$$\begin{aligned} &\pi_1(\alpha_X(t, x))(i) = \pi_1(t)(i) \\ &\pi_2(\alpha_X(t, x))(o) = \pi_2(t)(o) \\ &\pi_2(\alpha_X(t, x))(\delta) = \begin{cases} \{x\} & \text{if } \forall o \in O\colon \pi_2(t)(o) = \emptyset \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

In the above definition of α, we can see that we have captured what the deltafication should do. For input and output labels, the behavior of the resulting system is the same as for the original one. Only when we consider the special output label δ we see that only when there are no other output labels that are enabled, we add a self-loop to the behavior map. This is why we had to make sure that the state that we came from was remembered by the functor, otherwise there was no way to assign the singleton set of that state to the result after taking the δ transition. We show that α is indeed a natural transformation, so that we may use it to define deltafication.

Proposition 4.11. The above defined $\alpha\colon F \times \mathrm{id} \Rightarrow F_{\Delta}$ is a natural transformation.

Proof. Let $h\colon X \to Y$. We have to show that the following diagram commutes:

$$\begin{tikzcd} FX \times X \arrow[r, "\alpha_X"] \arrow[d, "Fh \times h"'] & F_{\Delta}X \arrow[d, "F_{\Delta}h"] \\ FY \times Y \arrow[r, "\alpha_Y"] & F_{\Delta}Y \end{tikzcd}$$

Consider $t \in FX$, $x \in X$, $a \in \Sigma$, and the label δ. Then we write out

$$\begin{aligned}
(F_\Delta h \circ \alpha_X)(t,x)(a) &= (F_\Delta h(\alpha_X((t,x)))(a) \\
&= F_\Delta h(\alpha_X((t,x))(a)) \\
&= F_\Delta h(t(a)) \\
&= (\mathcal{P}(h)(\pi_1(t)(a)), \mathcal{P}(h)(\pi_2(t)(a)) \\
&= \langle \mathcal{P}(h) \circ \pi_1(t), \mathcal{P}(h) \circ \pi_2(t)\rangle(a) \\
&= \alpha_Y(\langle \mathcal{P}(h) \circ \pi_1(t), \mathcal{P}(h) \circ \pi_2(t)\rangle, h(x))(a) \\
&= (\alpha_Y(Fh(t), h(x))(a) \\
&= (\alpha_Y(Fh \times h)((t,x))(a) \\
&= (\alpha_Y \circ (Fh \times h))(t,x)(a)
\end{aligned}$$

$$\begin{aligned}
(F_\Delta h \circ \alpha_X)(t,x)(\delta) &= (F_\Delta h(\alpha_X((t,x)))(\delta) \\
&= F_\Delta(h)(\alpha_X((t,x))(\delta)) \\
&= \begin{cases} \mathcal{P}(h)(\{x\}) & \text{if } \forall o \in O\colon t(o) = \emptyset \\ \mathcal{P}(h)(\emptyset) & \text{otherwise} \end{cases} \\
&= \begin{cases} \{h(x)\} & \text{if } \forall o \in O\colon \mathcal{P}(h)(t(o)) = \emptyset \\ \emptyset & \text{otherwise} \end{cases} \\
&= \alpha_Y(Fh(t), h(x))(\delta) \\
&= (\alpha_Y(Fh \times h)((t,x))(\delta) \\
&= (\alpha_Y \circ (Fh \times h))(t,x)(a)
\end{aligned}$$

□

Now that we have captured the construction, we have to piece it together with the functor that saves a copy of the relevant state. We use "id-α-lifting" as defined in definition 3.5 and then use functor composition with the functor $\mathrm{id}^\times$ we have already defined.

Definition 4.12. Deltafication is defined as $\Delta = \mathrm{id}_\alpha \circ \mathrm{id}^\times$. Taking a closer look, and filling in the details gives $\Delta\colon \mathsf{Coalg}(F) \to \mathsf{Coalg}(F_\Delta)$ where for an F-coalgebra (X, c) and an F-homomorphism h we have:

$$\begin{aligned}
\Delta((X,c)) &= (\mathrm{id}_\alpha \circ \mathrm{id}^\times)((X,c)) \\
&= \mathrm{id}_\alpha((X, \langle c, \mathrm{id}\rangle)) \\
&= (X, \alpha_X \circ \langle c, \mathrm{id}\rangle) \\
\Delta(h) &= (\mathrm{id}_\alpha \circ \mathrm{id}^\times)(h) = h
\end{aligned}$$

Example 4.13. Let $X = \{q_0, q_1, q_2\}$. Consider the F-coalgebra (X, c) for the map $c\colon X \to F(X)$ defined as follows.

$$\begin{array}{lclclclclcl}
c(q_0)(i) & = & \{q_1, q_2\} & c(q_1)(i) & = & \emptyset & c(q_2)(i) & = & \{q_2\} \\
c(q_0)(o) & = & \{q_2\} & c(q_1)(o) & = & \emptyset & c(q_2)(o) & = & \{q_2\}
\end{array}$$

If we now apply deltafication on (X, c) we get the F_Δ-coalgebra $\Delta((X, c))$ where the transition map is $(\alpha_X \circ \langle c, \mathrm{id} \rangle) \colon X \to F_\Delta(X)$. Working this out we get that the transitions in $\Delta((X, c))$ are defined as follows.

$$\begin{array}{lllll}
\alpha_X(c(q_0), q_0)(i) & = & c(q_0)(i) & = & \{q_1, q_2\} \\
\alpha_X(c(q_0), q_0)(o) & = & c(q_0)(o) & = & \{q_2\} \\
\alpha_X(c(q_0), q_0)(\delta) & = & c(q_0)(\delta) & = & \emptyset \\
\alpha_X(c(q_1), q_1)(i) & = & c(q_1)(i) & = & \emptyset \\
\alpha_X(c(q_1), q_1)(o) & = & c(q_1)(o) & = & \emptyset \\
\alpha_X(c(q_1), q_1)(\delta) & = & \{q_1\} & & \\
\alpha_X(c(q_2), q_2)(i) & = & c(q_2)(i) & = & \{q_2\} \\
\alpha_X(c(q_2), q_2)(o) & = & c(q_2)(o) & = & \{q_2\} \\
\alpha_X(c(q_2), q_2)(\delta) & = & c(q_2)(\delta) & = & \emptyset
\end{array}$$

This can be depicted as the following diagram, where after deltafication, the dashed δ-transition has been added.

$$\begin{tikzcd}
q_0 \arrow[r, "?i"] \arrow[dr, "{?i,!o}"'] & q_1 \arrow[loop right, dashed, "\delta"] \\
& q_2 \arrow[loop right, "{?i,!o}"]
\end{tikzcd}$$

4.3.2 Chaotic Completion

We have defined a way to capture quiescence in the traces of an LTS with the deltafication construction in the previous section. The next thing we want to capture in the traces is the behavior of traces that are underspecified by the specification. This is achieved by the construction of chaotic completion.

To define this construction we follow the same steps as we did for deltafication. We define an appropriate natural transformation, and then lift it to obtain chaotic completion. We recall the functor F_Δ and fix the functor $F_{\Delta,c}$. The subscript c for the functor $F_{\Delta,c}$ resembles the construction of chaotic completion.

$$\begin{array}{rcccc}
F_\Delta(X) & = & (I \to \mathcal{P}(X)) & \times & ((O + \{\delta\}) \to_{ne} \mathcal{P}(X)) \\
F_{\Delta,c}(X) & = & (I \to \mathcal{P}_{ne}(X)) & \times & ((O + \{\delta\}) \to_{ne} \mathcal{P}(X))
\end{array}$$

Note that we move from a possibly empty first projection, to a non-empty one. This resembles that for every state, and every input label, we can move to at least one state.

Let I be the set of inputs, and O be the set of outputs. Recall from definition 2.19 the chaotic LTS χ:

$$\begin{tikzcd}
{} \arrow[r] & q_\chi \arrow[loop above, "{I,O}"] \arrow[r, bend left, "\delta"] & q_\Omega \arrow[loop above, "\delta"] \arrow[l, bend left, "I"]
\end{tikzcd}$$

For an F_Δ-coalgebra, chaotic completion should add transitions to the chaotic state q_χ, just as done for LTSes in chapter 2. Also the transitions of the chaotic LTS χ should be added. We achieve this by adding the states using a functor, and then adding the transitions in the definition of the natural transformation.

Definition 4.14. We fix the set $2 = \{q_\chi, q_\Omega\}$. Then we consider the functor $(\_ + 2)\colon \mathsf{Set} \to \mathsf{Set}$, which adds the two states to a set X. Take the two functors $F_\Delta, F_{\Delta,c}\colon \mathsf{Set} \to \mathsf{Set}$ as defined above. Let $t \in F_\Delta$, $x \in X$, $i \in I$ and $o \in O$. We define the natural transformation $\gamma\colon (\_ + 2)F_\Delta \Rightarrow F_{\Delta,c}(\_ + 2)$, in terms of its components, as follows:

$$\begin{aligned}
\pi_1(\gamma_X(t))(i) &= \begin{cases} \pi_1(t(i)), & \text{if } \pi_1(t(i)) \neq \emptyset \\ \{q_\chi\}, & \text{else} \end{cases} \\
\pi_1(\gamma_X(q_\chi))(i) &= \{q_\chi\} \\
\pi_1(\gamma_X(q_\Omega))(i) &= \{q_\chi\} \\
\pi_2(\gamma_X(t))(o) &= \pi_2(t)(o) \\
\pi_2(\gamma_X(t))(\delta) &= \pi_2(t)(\delta) \\
\pi_2(\gamma_X(q_\chi))(o) &= \{q_\chi\} \\
\pi_2(\gamma_X(q_\chi))(\delta) &= \{q_\chi\} \\
\pi_2(\gamma_X(q_\Omega))(o) &= \emptyset \\
\pi_2(\gamma_X(q_\Omega))(\delta) &= \{q_\Omega\}
\end{aligned}$$

For input labels this natural transformation now adds transitions to q_χ if the input was not enabled. For output labels and for enabled input labels, the natural transformation does not alter the transition structure. For the newly added states q_χ and q_Ω we see that γ defines the transitions between them, following the diagram we have seen above.

We show that γ is indeed a natural transformation.

Proposition 4.15. The above defined $\gamma\colon (\_ + 2)F_\Delta \Rightarrow F_{\Delta,c}(\_ + 2)$ is a natural transformation.

Proof. Let $h\colon X \to Y$. We have to show that the following diagram commutes:

$$\begin{tikzcd}
(F_\Delta X) + 2 \arrow[r, "\gamma_X"] \arrow[d, "(F_\Delta h)+2"'] & F_{\Delta,c}(X + 2) \arrow[d, "F_{\Delta,c}(h+2)"] \\
(F_\Delta Y) + 2 \arrow[r, "\gamma_Y"] & F_{\Delta,c}(Y + 2)
\end{tikzcd}$$

Consider $t \in (F_\Delta X) + 2$, $x \in X$, $i \in I$, and $o \in O^\delta$. Then we write out

$$\begin{aligned}
\pi_1((F_{\Delta,c}(h+2)\circ\gamma_X))(t)(i) &= (\mathcal{P}(h+2)\circ\pi_1(\gamma_X))(t)(i)\\
&= \mathcal{P}(h+2)(\pi_1(\gamma_X(t)))(i)\\
&= \{(h+2)(t_1) \mid t_1 \in \pi_1(\gamma_X(t))(i)\}\\
&= \begin{cases} \{h(t_1(i)) \mid t_1 \in \pi_1(t(i))\} & \text{if } t \in F_\Delta X\\ \{q_X\} & \text{if } t \in 2 \end{cases}\\
&= \begin{cases} \pi_1((F_\Delta h)(t))(i)) & \text{if } t \in F_\Delta X\\ \{q_X\} & \text{otherwise} \end{cases}\\
&= \begin{cases} \pi_1(((F_\Delta h)+2)(t))(i)) & \text{if } ((F_\Delta h)+2)(t))(i) \neq \emptyset\\ \{q_X\} & \text{otherwise} \end{cases}\\
&= \pi_1(\gamma_Y((F_\Delta h)+2)(t))(i)\\
&= \pi_1((\gamma_Y \circ ((F_\Delta h)+2)))(t)(i)
\end{aligned}$$

$$\begin{aligned}
\pi_2((F_{\Delta,c}(h+2)\circ\gamma_X))(t)(o) &= (\mathcal{P}(h+2)\circ\pi_2(\gamma_X))(t)(o)\\
&= \mathcal{P}(h+2)(\pi_2(\gamma_X(t)))(o))\\
&= \{(h+2)(t_2) \mid t_2 \in \pi_2(\gamma_X(t))(o)\}\\
&= \begin{cases} \{h(t_2(o)) \mid t_2 \in \pi_2(t)(o)\} & \text{if } t \in F_\Delta X\\ \emptyset & \text{if } o \in O \wedge t = q_\Omega\\ \{t\} & \text{otherwise} \end{cases}\\
&= \begin{cases} \mathcal{P}(h)(\pi_2(t)(o)) & \text{if } t \in F_\Delta X\\ \emptyset & \text{if } o \in O \wedge t = q_\Omega\\ \{t\} & \text{otherwise} \end{cases}\\
&= \begin{cases} \pi_2(F_\Delta h(t))(o) & \text{if } t \in F_\Delta X\\ \emptyset & \text{if } o \in O \wedge t = q_\Omega\\ \{t\} & \text{otherwise} \end{cases}\\
&= \begin{cases} \pi_2(((F_\Delta h)+2)(t))(o)) & \text{if } t \in F_\Delta X\\ \emptyset & \text{if } o \in O \wedge t = q_\Omega\\ \{t\} & \text{otherwise} \end{cases}\\
&= \pi_2(\gamma_Y((F_\Delta h)+2)(t))(o)\\
&= \pi_2((\gamma_Y \circ ((F_\Delta h)+2)))(t)(o)
\end{aligned}$$

□

Chaotic completion is now defined as the "$(\_ + 2)$-γ-lifting".

Definition 4.16. Chaotic completion is defined as $\Xi = (\_ + 2)_\gamma$. Taking a closer look, and filling in the details gives $\Xi\colon \mathsf{Coalg}(F_\Delta) \to \mathsf{Coalg}(F_{\Delta,c})$ where for (X, c) an F_Δ-coalgebra and an F_Δ-homomorphism h we have:

$$\begin{aligned}\Xi((X,c)) &= (\_ + 2)_\gamma((X,c)) = (X + 2, \gamma_X \circ (c + 2))\\ \Xi(h) &= (\_ + 2)_\gamma(h) = (h + 2)\end{aligned}$$

Example 4.17. Again consider the LTS $\Delta((X, c))$ from example 4.13. For this system, chaotic completion adds the two states q_χ and q_Ω. It also adds the transitions for q_χ and q_Ω as they are defined in γ. For the other states, the only transition in $\Xi(\Delta((X, c))$ that is not the same as the matching transition in $\Delta((X, c))$ is the one for the state q_1 and the input label i. In $\Xi(\Delta((X, c))$ the transition structure is defined as $(\gamma_X \circ ((\alpha_X \circ \langle c, \mathrm{id}\rangle) + 2))\colon X \to F_{\Delta,c}(X)$.

$$\gamma_X(\alpha_X(c(q_1), q_1))(i) = \{q_\chi\}$$

Because the result of $\alpha_X(c(q_1), q_1)(i) = \emptyset$, we get that the above transition leads to the chaotic state q_χ. This can be depicted as the diagram below, where the dashed arrow is added after chaotic completion.

$$\begin{tikzcd}
q_0 \arrow[r, "?i"] \arrow[dr, "?i{,}!o"'] & q_1 \arrow[loop above, "\delta"] \arrow[r, dotted, "?i"] & q_\chi \arrow[loop above, "?i{,}!o"] \arrow[r, bend left, "\delta"] & q_\Omega \arrow[loop above, "\delta"] \arrow[l, bend left, "?i"] \\
 & q_2 \arrow[loop right, "?i{,}!o"] & &
\end{tikzcd}$$

4.3.3 Determinization

The last construction we define is determinization. This construction uses the powerset construction, which is a well-known construction from automata theory [14, 4]. It takes a non-deterministic system, and produces a deterministic one, ensuring that both systems yield the same traces. This is achieved by changing the state space to the powerset of the original state space, and making transitions move to the unions of states from the original system.

As mentioned in chapter 2, we want to consider deterministic systems to make sure the branching behavior of our system is captured in the traces. In the theory to come we will look at this behavior using final coalgebras. But to be able to look at a final coalgebra, it has to exist. For non-deterministic systems, this is not always the case. This is one of the reasons why we consider deterministic systems. A second reason is, that even if the final coalgebra exists for a non-deterministic system, it holds that behavioral equivalence coincides with strong bisimulation and not with trace equivalence. For deterministic systems, this notion of equivalence can actually be described as trace equivalence and behavior can be described using trace sets.

We recall the functor $F_{\Delta,c}$ and fix the functor $F_{\Delta,c,d}$. For the functors $F_{\Delta,c,d}$ the subscript d resembles the construction of determinization.

$$\begin{array}{rclcl} F_{\Delta,c}(X) & = & (I \to \mathcal{P}_{ne}(X)) & \times & ((O+\{\delta\}) \to_{ne} \mathcal{P}(X)) \\ F_{\Delta,c,d}(X) & = & (I \to X) & \times & ((O+\{\delta\}) \rightharpoonup_{ne} X) \end{array}$$

We now define a natural transformation that captures the construction. This definition is inspired by a similar construction we find in [14] were the same idea is worked out for non-deterministic automata.

Definition 4.18. Let $X \in \mathsf{Set}$. Let $T \in \mathcal{P}(F_{\Delta,c}(X))$ and $x \in X$. Then let $o \in (O+\{\delta\})$ and $i \in I$. Then the natural transformation $\beta\colon \mathcal{P}F_{\Delta,c} \Rightarrow F_{\Delta,c,d}\mathcal{P}$ is defined in terms of its projections, as follows:

$$\begin{array}{rcl} \pi_1(\beta_X(T))(i) & = & \displaystyle\bigcup_{t_1 \in \mathcal{P}(\pi_1)(T)} t_1(i) \\ \pi_2(\beta_X(T))(o) & = & \begin{cases} \displaystyle\bigcup_{t_2 \in \mathcal{P}(\pi_2)(T)} t_2(o) & \text{if } \exists t_2 \colon t_2(o) \neq \emptyset \\ \perp & \text{otherwise} \end{cases} \end{array}$$

In the natural transformation above we see that we define the result of a set of maps, as the union of the individual results. If none of the maps have a result, i.e. all of them map to the empty set, we define the new map as being undefined. For the first projection of the functor, $I \to \mathcal{P}_{ne}(X)$, we can see that any input for any state always leads to at least one state. That is why in the first projection of β, it can not happen that the result is undefined. We now show that $\beta\colon \mathcal{P}F_{\Delta,c} \Rightarrow F_{\Delta,c,d}\mathcal{P}$ defined above is indeed a natural transformation.

Proposition 4.19. The above defined $\beta\colon \mathcal{P}F_{\Delta,c} \Rightarrow F_{\Delta,c,d}\mathcal{P}$ is a natural transformation.

Proof. Let $h\colon X \to Y$. We have to show that the following diagram commutes.

$$\begin{tikzcd} \mathcal{P}(F_{\Delta,c}X) \arrow[r, "\beta_X"] \arrow[d, "\mathcal{P}(F_{\Delta,c}h)"'] & F_{\Delta,c,d}(\mathcal{P}X) \arrow[d, "F_{\Delta,c,d}(\mathcal{P}h)"] \\ \mathcal{P}(F_{\Delta,c}Y) \arrow[r, "\beta_Y"] & F_{\Delta,c,d}(\mathcal{P}Y) \end{tikzcd}$$

Consider $T \in \mathcal{P}(F_{\Delta,c}X)$, $x \in X$, $i \in I$ and $o \in \Sigma^\delta$. Then we write out

$$\begin{aligned} \pi_1(F_{\Delta,c,d}(\mathcal{P}h) \circ \beta_X)(T)(i) &= (\mathcal{P}h \circ \pi_1(\beta_X))(T)(i)) \\ &= \mathcal{P}h(\pi_1(\beta_X(T)))(i) \\ &= \{h(y) \mid y \in \pi_1(\beta_X(T))(i)\} \\ &= \{h(y) \mid y \in \bigcup_{t_1 \in \mathcal{P}(\pi_1)(T)} t_1(i))\} \end{aligned}$$

$$
\begin{aligned}
&= \bigcup_{t_1 \in \mathcal{P}(\pi_1)(T)} \mathcal{P}h(t_1(i)) \\
&= \bigcup_{t_1 \in \mathcal{P}(\pi_1)(T)} (F_{\Delta,c}h \circ t_1)(i) \\
&= \bigcup_{t_1' \in \mathcal{P}(\pi_1)\{(F_{\Delta,c}h)(t) \mid t \in T\}} t_1'(i) \\
&= \pi_1(\beta_Y(\{(F_{\Delta,c}h)(t) \mid t \in T\}))(i) \\
&= \pi_1(\beta_Y(\mathcal{P}(F_{\Delta,c}h)(T)))(i) \\
&= \pi_1(\beta_Y \circ \mathcal{P}(F_{\Delta,c}h))(T)(i)
\end{aligned}
$$

$$
\begin{aligned}
\pi_2(F_{\Delta,c,d}(\mathcal{P}h) \circ \beta_X)(T)(o) &= \mathcal{P}h(\pi_2(\beta_X(T)))(o) \\
&= \{h(y) \mid y \in \pi_2(\beta_X(T))(o)\} \\
&= \{h(y) \mid y \in \bigcup_{t_2 \in \mathcal{P}(\pi_2)(T)} t_2(o))\} \\
&= \begin{cases} \bigcup_{t_2 \in \mathcal{P}(\pi_2)(T)} (F_{\Delta,c}h \circ t_2)(o) & \text{if } \exists t_2' \colon t_2'(o) \neq \emptyset \\ \bot & \text{otherwise} \end{cases} \\
&= \begin{cases} \bigcup_{t_2' \in \mathcal{P}(\pi_2)\{(F_{\Delta,c}h)(t) \mid t \in T\}} t_2'(o) & \text{if } \exists t_2' \colon t_2'(o) \neq \emptyset \\ \bot & \text{otherwise} \end{cases} \\
&= \pi_2(\beta_Y(\{(F_{\Delta,c}h)(t) \mid t \in T\}))(o) \\
&= \pi_2(\beta_Y(\mathcal{P}(F_{\Delta,c}h)(T)))(o) \\
&= \pi_2(\beta_Y \circ \mathcal{P}(F_{\Delta,c}h))(T)(o)
\end{aligned}
$$

□

We define determinization as the "$\mathcal{P}$-β-lifting".

Definition 4.20. Determinization is defined as $\mathsf{det} = \mathcal{P}_\beta$. Taking a closer look, and filling in the details gives $\mathsf{det} \colon \mathsf{Coalg}(F_{\Delta,c}) \to \mathsf{Coalg}(F_{\Delta,c,d})$ where for an $F_{\Delta,c}$-coalgebra (X, c) and an $F_{\Delta,c}$-homomorphism h we have:

$$
\begin{aligned}
\mathsf{det}((X,c)) &= \mathcal{P}_\beta((X,c)) = (\mathcal{P}X, \beta_X \circ \mathcal{P}c) \\
\mathsf{det}(h) &= \mathcal{P}_\beta(h) = \mathcal{P}h
\end{aligned}
$$

Example 4.21. Consider the $F_{\Delta,c}$-coalgebra $\Xi(\Delta((X,c)))$ from example 4.17. The diagram for the $F_{\Delta,c,d}$-coalgebra we get after determinization is the following.

$$\begin{tikzcd}
 & & \{q_1\} \arrow[loop above, "\delta"] \arrow[r, "?i"] & \{q_\chi\} \arrow[loop above, "{?i,?o}"] \arrow[d, bend left, "\delta"] \\
\{q_0\} \arrow[r, "{?i,!o}"] & \{q_1, q_2\} \arrow[ur, "\delta"] \arrow[r, "?i"] \arrow[dr, "!o"] & \{q_2, q_\chi\} \arrow[loop above, "{?i,!o}"] \arrow[r, "\delta"] & \{q_\Omega\} \arrow[loop below, "\delta"] \arrow[u, bend left, "?i"] \\
 & & \{q_2\} \arrow[loop right, "{?i,!o}"] &
\end{tikzcd}$$

The traces that the states q_0, q_1 and q_2 could produce in the LTS $\Xi(\Delta((X, c)))$ are now exactly the traces produced by the states $\{q_0\}, \{q_1\}$ and $\{q_2\}$ in the determinized system. Also, the traces that appear in LTS $\Xi(\Delta((X, c)))$ are exactly the same as the ones that appear in $\mathsf{det}(\Xi(\Delta((X, c))))$. The main difference is that in $\mathsf{det}(\Xi(\Delta((X, c))))$ every trace has a unique path through the LTS.

4.4 Combining the Constructions

We have worked out the constructions deltafication, chaotic completion and determinization as functors between categories of coalgebras of specific types. The last remark we give in this section is showing how we will compose all of these constructions. During this chapter we have come across the following functors.

$$\begin{array}{lcrcl}
F(X) & = & (I \to \mathcal{P}(X)) & \times & (O \to \mathcal{P}(X)) \\
F_\Delta(X) & = & (I \to \mathcal{P}(X)) & \times & ((O + \{\delta\}) \to_{ne} \mathcal{P}(X)) \\
F_{\Delta,c}(X) & = & (I \to \mathcal{P}_{ne}(X)) & \times & ((O + \{\delta\}) \to_{ne} \mathcal{P}(X)) \\
F_{\Delta,c,d}(X) & = & (I \to X) & \times & ((O + \{\delta\}) \rightharpoonup_{ne} X)
\end{array}$$

We can obtain the uioco characterization after applying deltafication, chaotic completion and determinization in that order. This can be depicted as the diagram below.

$$\mathsf{Coalg}(F) \xrightarrow{\Delta} \mathsf{Coalg}(F_\Delta) \xrightarrow{\Xi} \mathsf{Coalg}(F_{\Delta,c}) \xrightarrow{\mathsf{det}} \mathsf{Coalg}(F_{\Delta,c,d})$$

The last step is considering the traces of the obtained system. We will show in the next section that this is a map that sends states of an LTS to the set of traces they produce.

Chapter 5

Final objects for LTSes

We have seen how to consider LTSes coalgebraically, and we have given the relevant LTS constructions. In this section we focus on the last component needed for the uioco characterization, namely the traces map. We do this by looking at the final object in the category of $F_{\Delta,c,d}$-coalgebras. We only consider trace sets for determinized LTSes. This is because for deterministic labeled transition systems, behavioral equivalence coincides with trace equivalence. This is close to behavioral equivalence for deterministic automata, where we know that it is described by language equivalence [9]. For non-deterministic systems, we do not get this, as behavioral equivalence for non-deterministic systems coincides with strong bisimulation [14, 13, 8].

We recall the functor

$$F_{\Delta,c,d}(X) = (I \to X) \times ((O + \{\delta\}) \rightharpoonup_{ne} X)$$

We also recall that we have defined the set of all labels as $\Sigma^\delta = I + O^\delta$ where $O^\delta = O + \{\delta\}$. We define an $F_{\Delta,c,d}$-coalgebra, where the states are trace sets and the transition structure is defined by trace derivatives, with the exception that the transition structure is undefined if the we get an empty derivative. We show that this coalgebra is the final object in the category of $F_{\Delta,c,d}$-coalgebras. The properties that we give to elements of the final coalgebra can be found in section 2.4. We first recall the definition of language derivative.

Definition 5.1. Let $L \subseteq (\Sigma^\delta)^*$, and let $a \in \Sigma^\delta$. Then the a-derivative of L is defined as

$$L_a = \{v \in (\Sigma^\delta)^* \mid av \in L\}$$

Definition 5.2. We define the $F_{\Delta,c,d}$-coalgebra $(\mathcal{L}_{\text{pin}}, d)$ where

$$\mathcal{L}_{\text{pin}} = \{\ L \subseteq (\Sigma^\delta)^* \quad | \quad \begin{array}{l} L \text{ is non-empty, prefix closed,} \\ \text{input enabled and non-blocking } \} \end{array}$$

and the map $d\colon \mathcal{L}_{\text{pin}} \to (I \to \mathcal{L}_{\text{pin}}) \times ((O + \{\delta\}) \rightharpoonup_{ne} \mathcal{L}_{\text{pin}})$ is defined such that for $L \in \mathcal{L}_{\text{pin}}$, $i \in I$, $o \in O^\delta$ we have

$$\begin{aligned} \pi_1(d(L))(i) &= L_i \\ \pi_2(d(L))(o) &= \begin{cases} L_o & \text{if } L_o \neq \emptyset \\ \perp & \text{otherwise} \end{cases} \end{aligned}$$

Remark 5.3. For $L \in \mathcal{L}_{\text{pin}}$ we have that $\varepsilon \in L$. This is because L is non-empty and prefix closed, and ε is a prefix of any trace.

To show the $F_{\Delta,c,d}$-coalgebra $(\mathcal{L}_{\text{pin}}, d)$ is indeed a final coalgebra, we first show that the map d is well typed. Then we show that for any $F_{\Delta,c,d}$-coalgebra there exists an $F_{\Delta,c,d}$-homomorphism into $\mathcal{L}_{\text{pin}}$, and that if there exist two, that they must be the same.

Proposition 5.4. The map $d\colon \mathcal{L}_{\text{pin}} \to (I \to \mathcal{L}_{\text{pin}}) \times ((O + \{\delta\}) \rightharpoonup_{ne} \mathcal{L}_{\text{pin}})$ is well typed.

Proof. Let $L \in \mathcal{L}_{\text{pin}}$. Let $a \in \Sigma^\delta$. We need to prove that for any $L \in \mathcal{L}_{\text{pin}}$ it holds that $d(L) \in F_{\Delta,c,d}(\mathcal{L}_{\text{pin}})$. This means that we need to check that the result of applying $d(L)$ to a label a is a non-empty, prefix closed, input enabled, non-blocking language. The only exception is that the type of $F_{\Delta,c,d}(\mathcal{L}_{\text{pin}})$ allows the second projection $\pi_2(d(L))(a)$, to be undefined. We prove each of the required properties individually, and make a case distinction whether $a \in I$ or $a \in O^\delta$.

Non-empty:
<ins>Case $a \in I$</ins>:
We have that $\pi_1(d(L))(a) = L_a$. Since L is input enabled, and because we know that $\varepsilon \in L$ from 5.3, we get that $a \in L$. This gives that $\varepsilon \in L_a$, by the definition of derivative. Hence L_a is non-empty.
<ins>Case $a \in O^\delta$</ins>:
If $\pi_2(d(L))(a) = \perp$, then d is of the right type, as the second projection of $d(L)$ is a partial map. This means that it is allowed to be undefined. If $\pi_2(d(L))(a) = L_a$, then we get from the definition of d that L_a is non-empty.

Prefix closed:
<ins>Case $a \in I$</ins>:
We have that $\pi_1(d(L))(a) = L_a$. Let $\sigma \in (\Sigma^\delta)^*$ and $l \in \Sigma^\delta$ such that $\sigma l \in L_a$. From the definition of derivative, we get that $a\sigma l \in L$. Since L is prefix closed, we get that $a\sigma \in L$. Then using the definition of derivative again we get that $\sigma \in L$ as needed.

Case $a \in O^\delta$:
If $\pi_2(d(L))(a) = \bot$, then d is of the right type, as the second projection of $d(L)$ is a partial map. This means that it is allowed to be undefined. If $\pi_2(d(L))(a) = L_a$, then the proof matches the other case.

Input enabled:
The proof is similar to the proof for prefix closed.

Non-blocking:
The proof is similar to the proof for prefix closed. □

Just like in section 2.1 we define the traces map for LTSes based on the inductively defined transition relation from definition 4.7. For $F_{\Delta,c,d}$-coalgebras we clarify what this relation is by giving the matching definition. The main difference for $F_{\Delta,c,d}$-coalgebras is that the transition relation only produces a single state as an output. For F-coalgebras the result was a set of states.

Definition 5.5. Let (X, c) be an $F_{\Delta,c,d}$-coalgebra. Let $x, x' \in X$ and let $a \in \Sigma^\delta$. We inductively define the relation $\rightarrow \subseteq X \times \Sigma^* \times X$ with the rules:

$$\frac{}{x \xrightarrow{\varepsilon} x}\ (1) \qquad \frac{\exists x'' \in X : c(x)(a) = x'' \quad x'' \xrightarrow{\sigma} x'}{x \xrightarrow{a\sigma} x'}\ (2)$$

We can now give the definition of the map $\mathsf{traces}\colon X \rightarrow \mathcal{L}_{\text{pin}}$ and prove that it is an $F_{\Delta,c,d}$-homomorphism.

Definition 5.6. Let (X, c) be an $F_{\Delta,c,d}$-coalgebra, and let $(\mathcal{L}_{\text{pin}}, d)$ be the $F_{\Delta,c,d}$-coalgebra defined above. Let $a \in \Sigma^\delta$ and $\sigma \in (\Sigma^\delta)^*$. Then we define the map $\mathsf{traces}\colon X \rightarrow \mathcal{L}_{\text{pin}}$ such that for any $x \in X$ we have

$$\sigma \in \mathsf{traces}(x) \Leftrightarrow \exists x' \in X : x \xrightarrow{\sigma} x'$$

We first show that traces is well-typed, i.e. for any $F_{\Delta,c,d}$ coalgebra traces sends each state to a prefix closed, input enabled, non-blocking set of traces.

Proposition 5.7. The map traces is well-typed.

Proof. Let (X, c) be an $F_{\Delta,c,d}$-coalgebra. Let $x \in X$. We need to prove that $\mathsf{traces}(x)$ is non-empty, prefix closed, input enabled and non-blocking.

Non-empty:
From definition 4.7.1 we get that $\varepsilon \in \mathsf{traces}(x)$, hence it is non-empty.

Prefix-closed:
Let $\sigma \in \Sigma^*$ and let $l \in \Sigma^\delta$. Assume $\sigma l \in \mathsf{traces}(x)$. We need to show that $\sigma \in \mathsf{traces}(x)$. We do this by induction on σ.

Base case:
We take $\sigma = \varepsilon$. Assume $a \in \mathsf{traces}(x)$. Then by definition 4.7.1 we get that indeed $\varepsilon \in \mathsf{traces}(x)$.

Induction hypothesis:
Let $\sigma \in \Sigma^*$ and $\sigma' \in \Sigma^*$. Let $l \in \Sigma^\delta$. We assume that $\sigma l \in \mathsf{traces}(x) \implies \sigma \in \mathsf{traces}(x)$.

Induction step:
Let $\sigma \in \Sigma^*$ and let $a \in \Sigma^\delta$. We need to show that if $a\sigma l \in \mathsf{traces}(x)$, that then also $a\sigma \in \mathsf{traces}(x)$. Assume $a\sigma l \in \mathsf{traces}(x)$. From the definition of traces we get that $\exists x' \in X$ such that $x \xrightarrow{a\sigma l} x'$. From definition 4.7.2 we the get that $\exists x'' \in X$ such that $c(x)(a) = x''$ and that $x'' \xrightarrow{\sigma l} x'$. From the definition of traces this gives that $\sigma l \in \mathsf{traces}(x'')$. We can now use the induction hypothesis to get that $\sigma \in \mathsf{traces}(x'')$. From the definition of traces we now know that $\exists x''' \in X$ such that $x'' \xrightarrow{\sigma} x'''$. Combining this with the fact that $c(x)(a) = x''$, definition 4.7.2 gives us that $x \xrightarrow{a\sigma} x'''$. From the definition of traces we may now conclude that $a\sigma \in \mathsf{traces}(x)$ as needed.

Input enabled:
Let $\sigma \in \mathsf{traces}(x)$ and let $i \in I$. We need to prove that $\sigma i \in \mathsf{traces}(x)$. We do this by induction on σ.

Induction base:
Let $\sigma = \varepsilon$. From the type of the functor $F_{\Delta,c,d}$ we get that the first projection is a total function. That we know that for any state $x \in X$ and any input label $i \in I$ we know that there exists $x' \in X$ such that $c(x)(i) = x'$. From definition 4.7.1 we get that $x' \xrightarrow{\varepsilon} x'$. We combine these two to get from definition 4.7.2 that $x \xrightarrow{i} x'$. It then follows from the definition of traces that $i \in \mathsf{traces}(x)$.

Induction hypothesis:
Let $\sigma \in \Sigma^*$ and let $i \in I$. We assume that $\sigma \in \mathsf{traces}(x) \implies \sigma i \in \mathsf{traces}(x)$.

Induction step:
Let $\sigma \in \Sigma^*$ and $a \in \Sigma^\delta$ such that $a\sigma \in \mathsf{traces}(x)$. Let $i \in I$. We need to show that $\sigma a i \in \mathsf{traces}(x)$. Assume $a\sigma \in \mathsf{traces}(x)$. From the definition of traces we get that $\exists x' \in X$ such that $x \xrightarrow{a\sigma} x'$. Then definition 4.7.2 gives that $\exists x'' \in X$ with $c(x)(a) = x''$ and $x'' \xrightarrow{\sigma} x'$. From this we see that $\sigma \in \mathsf{traces}(x'')$. We may now use the induction hypothesis to get that $\forall i \in I$ we have $\sigma i \in \mathsf{traces}(x'')$. The definition of traces then gives that $\exists x''' \in X$ such that $x'' \xrightarrow{\sigma i} x'''$. Together with the fact that $c(x)(a) = x''$ definition 4.7.2 then gives us that $x \xrightarrow{a\sigma i} x'''$. Thus we can conclude that $a\sigma i \in \mathsf{traces}(x)$ as needed.

Non-blocking:
Let $\sigma \in \mathsf{traces}(x)$. We need to prove that $\exists o \in O + \{\delta\}$ such that $\sigma o \in \mathsf{traces}(x)$. We do this by induction on σ.

Induction base:
Let $\sigma = \varepsilon$. Then we have to show that exists $o \in O + \{\delta\}$ such that $o \in \mathsf{traces}(x)$. From the type of the functor $F_{\Delta,c,d}$ we get that the second projection is a non-empty function from $O + \{\delta\}$ to X, meaning that there exists at least one

$o \in O + \{\delta\}$ such that $\exists x' \in X$ with $c(x)(o) = x'$. We know from definition 4.7.1 that $x' \xrightarrow{\varepsilon} x'$. Combining these two, we get from definition 4.7.2 that $x \xrightarrow{o} x'$. Then the definition of traces gives us that $o \in \mathsf{traces}(x)$, as needed.

Induction hypothesis:
Let $\sigma \in \Sigma^*$. We assume that $\exists o \in O + \{\delta\}$ such that $\sigma o \in \mathsf{traces}(x)$.

Induction step:
Let $\sigma \in \Sigma^*$ and $a \in \Sigma^\delta$. Assume $\sigma \in \mathsf{traces}(x)$. We need to show that $\exists o \in O + \{\delta\}$ such that $a\sigma o \in \mathsf{traces}(x)$. From the definition of traces we get that $\exists x' \in X$ such that $x \xrightarrow{a\sigma} x'$. Then from definition 4.7.2 we get that $\exists x'' \in X$ such that $c(x)(a) = x''$ and that $x'' \xrightarrow{\sigma}$. From the definition of traces we get that $\sigma \in \mathsf{traces}(x'')$. We may now use the induction hypothesis to get that $\exists o \in O + \{\delta\}$ such that $\sigma o \in \mathsf{traces}(x'')$. The definition of traces gives that $\exists x''' \in X$ with $x'' \xrightarrow{\sigma o} x'''$. From this, together with the fact that $c(x)(a) = x''$, we get from definition 4.7.2 that $x \xrightarrow{a\sigma o} x''$. Using the definition of traces, we get that $a\sigma o = \sigma a o \in \mathsf{traces}(x)$ as needed. □

The next thing we prove is that traces is indeed an $F_{\Delta,c,d}$-homomorphism. In order to achieve this we need the following lemma for one of the intermediate steps in the proof.

Lemma 5.8. Consider an $F_{\Delta,c,d}$-coalgebra (X, c). Let $x, y \in X$ and $a \in \Sigma^\delta$ such that $x \xrightarrow{a} y$. Then it holds that

$$\mathsf{traces}(x)_a = \mathsf{traces}(y)$$

Proof. Take an arbitrary $\sigma \in (\Sigma^\delta)^*$. We can then see that

$$\begin{aligned}
\sigma \in \mathsf{traces}(x)_a \quad &\Leftrightarrow \quad a\sigma \in \mathsf{traces}(x) && \text{(def derivative)}\\
&\Leftrightarrow \quad \exists y' \in X : x \xrightarrow{a\sigma} y' && \text{(def } \mathsf{traces}\text{)}\\
&\Leftrightarrow \quad \exists y'' \in X : c(x)(a) = y'' \wedge y'' \xrightarrow{\sigma} y' && \text{(def 4.7.2)}\\
&\Leftrightarrow \quad \exists y'' \in X : x \xrightarrow{a} y'' \wedge y'' \xrightarrow{\sigma} y' && \text{(def 4.7.2)}\\
&\Leftrightarrow \quad x \xrightarrow{a} y \wedge y \xrightarrow{\sigma} y' && (*)\\
&\Leftrightarrow \quad x \xrightarrow{a} y \wedge \sigma \in \mathsf{traces}(y) && \text{(def } \mathsf{traces}\text{)}
\end{aligned}$$

$(*)$: we use that the $F_{\Delta,c,d}$-coalgebra (X, c) is deterministic. □

Using this lemma, we can now prove that traces is an $F_{\Delta,c,d}$-homomorphism.

Proposition 5.9. The map traces is an $F_{\Delta,c,d}$-homomorphism.

Proof. We fill in the definition of a homomorphism for coalgebras. From this we get that the map $\mathsf{traces}\colon X \to \mathcal{L}_{\text{pin}}$ is an $F_{\Delta,c,d}$-homomorphism if it makes the following square commute:

$$\begin{tikzcd}
X \arrow[r, "\mathsf{traces}"] \arrow[d, "c"'] & \mathcal{L}_{\text{pin}} \arrow[d, "d"] \\
(I \to X) \times ((O + \{\delta\}) \rightharpoonup_{ne} X) \arrow[r, "F_{\Delta,c,d}(\mathsf{traces})"] & (I \to \mathcal{L}_{\text{pin}}) \times ((O + \{\delta\}) \rightharpoonup_{ne} \mathcal{L}_{\text{pin}})
\end{tikzcd}$$

We get this by showing that the following equation holds for arbitrary $x \in X$ and $a \in \Sigma^\delta$.

$$(d \circ \mathsf{traces})(x)(a) = (F_{\Delta,c,d}(\mathsf{traces}) \circ c)(x)(a)$$

We can derive the following:

$$\begin{aligned}
(d \circ \mathsf{traces})(x)(a) &= d(\mathsf{traces}(x))(a) \\
&= \begin{cases} \mathsf{traces}(x)_a, & \text{if } \mathsf{traces}(x)_a \neq \emptyset \\ \bot, & \text{otherwise} \end{cases} && \text{(def } d\text{, Remark 4.6)} \\
&= \begin{cases} \mathsf{traces}(x)_a, & \text{if } a \in \mathsf{traces}(x) \\ \bot, & \text{otherwise} \end{cases} && (*) \\
&= \begin{cases} \mathsf{traces}(x)_a, & \text{if } \exists y \in X \colon x \xrightarrow{a} y \\ \bot, & \text{otherwise} \end{cases} && \text{(def } \mathsf{traces}\text{)} \\
&= \begin{cases} \mathsf{traces}(y), & \text{if } \exists y \in X \colon x \xrightarrow{a} y \\ \bot, & \text{otherwise} \end{cases} && \text{(Lemma 5.8)} \\
&= (\mathsf{traces} + \mathrm{id})(c(x)(a)) \\
&= F_{\Delta,c,d}(\mathsf{traces})(c(x))(a) \\
&= (F_{\Delta,c,d}(\mathsf{traces}) \circ c)(x)(a)
\end{aligned}$$

$(*)$: Since $\mathsf{traces}_a \neq \emptyset$ we have that there exists some $\sigma \in (\Sigma^\delta)^*$ such that $a\sigma \in \mathsf{traces}(x)$. Since $\mathsf{traces}(x) \in \mathcal{L}_{\text{pin}}$, we have that $\mathsf{traces}(x)$ is prefix closed, hence $a \in \mathsf{traces}(x)$. □

We have seen that traces is an $F_{\Delta,c,d}$-homomorphism. We now want to show that it is unique. We do this by showing that if we have another $F_{\Delta,c,d}$-homomorphism $h\colon X \to \mathcal{L}_{\text{pin}}$, that is is equal to traces.

Proposition 5.10. Let $h\colon X \to \mathcal{L}_{\text{pin}}$ be an $F_{\Delta,c,d}$-homomorphism. Then it holds that $h = \mathsf{traces}$.

Proof. We show by induction on $\sigma \in (\Sigma^\delta)^*$ that for any $F_{\Delta,c,d}$-homomorphism $h : X \to \mathcal{L}_{\text{pin}}$ and any $x \in X$ we have that $\sigma \in h(x) \Leftrightarrow \sigma \in \mathsf{traces}(x)$. We need to show that h satisfies both the properties from the definition of traces.

Induction base:
Let $\sigma = \varepsilon$. Then $\varepsilon \in h(x)$ and $\varepsilon \in \mathsf{traces}(x)$ by 5.3, as $h(x) \in \mathcal{L}_{\text{pin}}$ and $\mathsf{traces}(x) \in \mathcal{L}_{\text{pin}}$.

Induction hypothesis:
Let $\sigma \in (\Sigma^\delta)^*$. Then assume that $\sigma \in h(x) \Leftrightarrow \sigma \in \mathsf{traces}(x)$.

Induction step:
Let $\sigma \in (\Sigma^\delta)^*$ and let $a \in \Sigma^\delta$. Then

$$\begin{array}{rcll}
a\sigma \in h(x) & \Leftrightarrow & \sigma \in h(x)_a & \text{(def derivative)} \\
& \Leftrightarrow & \sigma \in d(h(x))(a) & \text{(def } d\text{)} \\
& \Leftrightarrow & \sigma \in F_{\Delta,c,d}(h)(c(x))(a) & (h \text{ an } F_{\Delta,c,d}\text{-hom}) \\
& \Leftrightarrow & \sigma \in (h + \mathrm{id})(c(x)(a)) & \\
& \Leftrightarrow & \sigma \in h(y) & (*) \\
& \Leftrightarrow & \sigma \in \mathsf{traces}(y) & \text{(IH)} \\
& \Leftrightarrow & \sigma \in (\mathsf{traces} + \mathrm{id})(c(x)(a)) & \\
& \Leftrightarrow & \sigma \in F_{\Delta,c,d}(\mathsf{traces})(c(x))(a) & \\
& \Leftrightarrow & \sigma \in d(\mathsf{traces}(x))(a) & (\mathsf{traces} \text{ an } F_{\Delta,c,d}\text{-hom}) \\
& \Leftrightarrow & \sigma \in \mathsf{traces}(x)_a & \text{(def } \mathsf{traces}\text{)} \\
& \Leftrightarrow & a\sigma \in \mathsf{traces}(x) & \text{(def derivative)}
\end{array}$$

$(*)$: We know that $\exists y \in X$ such that $c(x)(a) = y$, otherwise the assumption $a\sigma \in h(x)$ would not hold. □

Now that we have shown that the traces map we have defined above is a unique $F_{\Delta,c,d}$-homomorphism that maps any $F_{\Delta,c,d}$-coalgebra to $(\mathcal{L}_{\text{pin}}, d)$, we may conclude that $(\mathcal{L}_{\text{pin}}, d)$ is indeed the final object in the category of $F_{\Delta,c,d}$-coalgebras.

Corollary 5.11. The final object of $\mathsf{Coalg}(F_{\Delta,c,d})$ is $(\mathcal{L}_{\text{pin}}, d)$.

This now leaves us with the fact that the traces map maps any state in a non-blocking, input enabled, deterministic coalgebra to a trace set that is non-empty, prefix closed, input enabled and non-blocking. Also, since the traces map is the behavior map for $F_{\Delta,c,d}$-coalgebras, and because it maps any state to the trace set it produces, we get that for $F_{\Delta,c,d}$-coalgebra, behavioral equivalence is matched by equality of trace sets.

Chapter 6

Suspension Language Properties, Coalgebraically

In this section we will look at the properties of being anomaly free, quiescence reducible and quiescence stable. We formalize these properties in terms of states of $F_{\Delta,c,d}$-coalgebras. We then prove that the constructions we have defined in section 4.3 lead to coalgebras that are anomaly-free, quiescence-reducible and quiescence-stable. Lastly we look at the subcategory of $F_{\Delta,c,d}$-coalgebras that are anomaly-free, quiescence-reducible and quiescence-stable, which we call suspension coalgebras, and show that the final suspension coalgebra is a restriction of the final $F_{\Delta,c,d}$-coalgebra.

In section 2.4 we have briefly seen what a suspension language is. We recall the definition here.

Definition 6.1. Let $\Sigma^\delta = I + O^\delta$ be a set of labels. A set $L \subseteq (\Sigma^\delta)^*$ is

Prefix closed	$\Leftrightarrow$	$\forall \sigma \in (\Sigma^\delta)^*, \forall l \in \Sigma^\delta \colon \sigma l \in L \implies \sigma \in L$
Non-blocking	$\Leftrightarrow$	$\forall \sigma \in L,\ \exists o \in O^\delta \colon \sigma o \in L$
Input enabled	$\Leftrightarrow$	$\forall \sigma \in L,\ \forall i \in I \colon \sigma i \in L$
Anomaly free	$\Leftrightarrow$	$\forall \sigma \in L,\ \neg\exists \sigma_1, \sigma_2 \in (\Sigma^\delta)^*,\ \neg\exists o \in O \colon \sigma = \sigma_1 \delta o \sigma_2$
Quiescence reducible	$\Leftrightarrow$	$\forall \sigma, \rho \in (\Sigma^\delta)^* \colon \sigma\delta\rho \in L \implies \sigma\rho \in L$
Quiescence stable	$\Leftrightarrow$	$\forall \sigma, \rho \in (\Sigma^\delta)^* \colon \sigma\delta\rho \in L \implies \sigma\delta\delta\rho \in L$

If $L \subseteq (\Sigma^\delta)^*$ is non-empty and satisfies all of the above properties, L is called a suspension language.

6.1 Definitions

The properties we formalize describe the behavior we expect for the transition systems we obtain after applying the constructions mentioned in 4.3. These properties all reason about traces that include δ. In short, the properties ensure three things: that we do not see output after quiescence, that we can leave quiescence out of a trace and that we can duplicate quiescence once we have observed it. We define the properties for F_Δ-coalgebras, as those are the first flavor of coalgebra that consider the label δ. For the functors $F_{\Delta,c}$ and $F_{\Delta,c,d}$ the definitions do not change much, as they are defined using the transition relation "$\rightarrow$". However, it can happen that the behavior map on which the transition relation is defined changes, as the functor of the LTS at hand changes. It should be clear from context, what the exact rules of the transition relation are, once we encounter them.

We first give definitions for all of the properties we want to investigate. Then we show that applying the construction of deltafication, chaotic completion and determinization ensures that the properties hold.

6.1.1 Anomaly Free

The first property we describe is the property of being anomaly free. An anomaly is behavior that is deemed abnormal, or impossible. In this case that means that we find the following behavior: we can first make a step with the label δ, and afterwards we can make a step with an output label. This is then an anomaly, as the label δ should indicate the absence of output, and taking the δ-transition should not make it so that any output becomes enabled.

Definition 6.2. Let (X, c) be an F_Δ-coalgebra. We say (X, c) is anomaly free if $\forall x, y \in X, \forall o \in O$ we have

$$x \xrightarrow{\delta} y \implies y \overset{o}{\not\rightarrow}$$

Example 6.3. The following LTS is anomaly free.

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "o"] \arrow[dr, "i"'] & q_1 \arrow[loop above, "\delta"] \arrow[r, "i"] & q_2 \arrow[loop above, "\delta"] \\
& & q_3 \arrow[ur, "o"'] &
\end{tikzcd}$$

We can see this because for the states where the label δ is enabled, which is the case in q_1 and q_2, there are not output labels enabled.

6.1.2 Quiescence Reducible

Then we look at the property of being quiescence reducible. This property states that taking a δ-transition should always be optional. For an LTS this means that if we take a transition with the label δ and then continue with some trace, we could have also chosen to not take the δ-transition and still be able to continue with the same trace. In the definition of quiescence reducibility, we use that the states before the δ-transition should simulate the state after it. We first give a definition of simulation for F_Δ-coalgebras.

Definition 6.4. Let (X, c) be an F_Δ-coalgebra. A simulation is a relation $R \subseteq X \times X$ such that $\forall (x, y) \in R, \forall a \in \Sigma^\delta$ we have

$$\forall x' \in X : x \xrightarrow{a} x' \implies \exists y' \in X : y \xrightarrow{a} y' \wedge (x', y') \in R$$

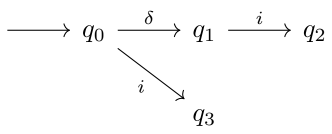
For $x, y \in X$ we have that if $(x, y) \in R$ we say that y simulates x, or that x is simulated by y. The simulation that contains all simulations, i.e. the greatest simulation, is denoted by $\lesssim$.

We now give the definition for quiescence reducibility for LTSes.

Definition 6.5. Let (X, c) be an F_Δ-coalgebra. We say (X, c) is quiescence reducible if $\forall x, y \in X$ we have

$$y \xrightarrow{\delta} x \implies x \lesssim y$$

Example 6.6. The following LTS is quiescence reducible.


$$\begin{tikzcd} {} \arrow[r] & q_0 \arrow[r, "\delta"] \arrow[dr, "i"'] & q_1 \arrow[r, "i"] & q_2 \\ & & q_3 & \end{tikzcd}$$

From q_0 the only trace involving δ is δi. We can see that leaving out the quiescence also gives a trace that starts at q_0.

6.1.3 Quiescence Stable

Lastly we consider quiescence stability. If an LTS is quiescence stable it holds that whenever we take a δ-transition, it should be possible to take another one. After we take the second δ-transition, we should be able to continue with the same traces as we were able to, before we took it. This captures that taking a δ-transition can be done as many times as we want, without changing the options we have afterwards. The definition again uses a simulation.

Definition 6.7. Let (X, c) be an F_Δ-coalgebra. We say (X, c) is quiescence stable if $\forall x, y \in X$ we have

$$x \xrightarrow{\delta} y \implies \exists y', y'' \in X : x \xrightarrow{\delta} y' \wedge y' \xrightarrow{\delta} y'' \wedge y'' \lesssim y$$

We point out that in the above definition we take into account that in an arbitrary F_Δ-coalgebra, we might have non-determinism for δ transitions. That is why we use the states y' and y'', instead of just y. The F_Δ-coalgebras we will come across are the ones we obtain after applying deltafication. For such systems it holds that δ labels only appear on loops. That is why in the proofs to come, we can often use the state y for both y' and y''.

Example 6.8. The following LTS is quiescence stable.

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "o"] & q_1 \arrow[r, "i"] \arrow[loop above, "\delta"] & q_2
\end{tikzcd}$$

For every trace involving δ, we can see that doubling it also gives a trace of the LTS. This is because of the δ loop on q_1.

6.2 Checking the constructions

We now show that the LTS constructions we have described in 4.3 lead to LTSes which satisfy the properties above. We do this by considering the constructions one at a time, and showing that the properties hold for the intermediate LTSes we obtain. As we see, after only applying deltafication, we are left with an LTS which is already anomaly free, quiescence stable and quiescence reducible. This means that for chaotic completion and determinization we have to show that they preserve the three properties.

Proposition 6.9. Let (X, c) be an F-coalgebra. Then $\Delta((X, c))$ is anomaly free, quiescence stable and quiescence reducible.

Proof. We first write out $\Delta((X, c)) = (X, \alpha_X \circ \langle c, \mathrm{id}\rangle)$. We now prove the properties separately.

Anomaly free:
Let $x, y \in X$ such that $x \xrightarrow{\delta} y$, We have to prove that $\forall o \in O$ it holds that $y \overset{o}{\not\to}$. Since $x \xrightarrow{\delta} y$ we get from definition 4.7.2 that $y \in (\alpha_X \circ \langle c, \mathrm{id}\rangle)(x)(\delta)$. We write out:

$$\begin{array}{rcll} (\alpha_X \circ \langle c, \mathrm{id}\rangle)(x)(\delta) & = & \alpha_X(c(x), x)(\delta) & \\ & = & \{x\} & (*) \end{array}$$

$(*)$: Since $y \in (\alpha_X \circ \langle c, \mathrm{id}\rangle)(x)(\delta)$, we have that $\alpha_X(c(x), x)(\delta) \neq \emptyset$.

As $\alpha_X(c(x), x)(\delta) = \{x\}$ we get that $\forall o \in O$ we have $x \overset{o}{\not\to}$. From the derivation we get that $y \in \{x\}$, hence $y = x$. So indeed it holds that $\forall o \in O$ we have $y \overset{o}{\not\to}$.

Quiescence reducible:
Let $x, y \in X$ such that $x \xrightarrow{\delta} y$. We have to show that $x \lesssim y$. Because we know $x \xrightarrow{\delta} y$, we get $(\alpha_X \circ \langle c, \mathrm{id}\rangle)(x)(\delta) = \{x\}$, hence $y = x$. Now we get that $x \lesssim y$ with the simulation $R = \{(x, x) \mid x \in X\}$.

Quiescence stable:
Let $x, y \in X$ such that $x \xrightarrow{\delta} y$. We have to show that $\exists y', y'' \in X$ such that $x \xrightarrow{\delta} y'$, $y' \xrightarrow{\delta} y''$ and $y'' \lesssim y$. As we have seen, for the transition $x \xrightarrow{\delta} y$, we get that $y = x$. So, we take $y = y' = y'' = x$. This gives that $x \xrightarrow{\delta} y'$, $y' \xrightarrow{\delta} y''$. Again using the simulation $R = \{(x, x) \mid x \in X\}$ we get $y'' \lesssim y$. □

The next construction we will check is chaotic completion. We want to show that given an anomaly free, quiescence stable, quiescence reducible F_Δ-coalgebra, it holds that chaotic completion preserves these properties. However, for quiescence reducibility this does not hold immediately. Consider a quiescence stable F_Δ-coalgebra (X, c) where for $x, y \in X$ it holds that $x \xrightarrow{\delta} y$. Then because (X, c) is quiescence stable it holds that x simulates y, i.e. all of the traces that can be produced from y, can also be produced from x. Now it can happen that there is an input label $i \in I$ that is not enabled in y, but is enabled in x. Then chaotic completion would add an i-transition from y to the state q_χ, and in turn add traces to the trace set produced by y, that can not be matched by the trace set produced by x. This would mean that after chaotic completion, x no longer simulates y. The same can happen for quiescence stability. To solve this problem, we use that we know that the F_Δ-coalgebras we consider are obtained from applying deltafication on F-coalgebras.s From this we know that all the delta transitions that appear are self loops.

Proposition 6.10. Let (X, c) be an anomaly free, quiescence reducible, quiescence stable F_Δ-coalgebra, such that $(X, c) = \Delta((Y, b))$ for some F-coalgebra (Y, b). Then $\Xi((X, c))$ is anomaly free, quiescence reducible and quiescence stable.

Proof. Let (X, c) be an anomaly free, quiescence reducible, quiescence stable F_Δ-coalgebra, such that $(X, c) = \Delta((Y, b))$ for some F-coalgebra (Y, b). We first write out $\Xi((X, c)) = (X + 2, \gamma_X \circ (c + 2))$. The chaotic states that are added are anomaly free, quiescence reducible and quiescence stable per construction. Also note that the only way to get a transition to a chaotic state, is by considering an input label.

We now have to show that for $x \in X$ the properties also hold. We write out

$$\begin{aligned} (\gamma_X \circ (c + 2))(x)(\delta) &= \gamma_X(c(x))(\delta) \\ &= c(x)(\delta) \end{aligned}$$

From this we can see that for δ the transitions in $\Xi((X, c))$ move to the same states as the transitions in (X, c). We now prove the properties separately.

Anomaly free:
Let $x, y \in X$ such that $x \xrightarrow{\delta} y$. Using definition 4.7.2 we can see that we have $y \in (\gamma_X \circ (c + 2))(x)(\delta)$. We have also seen that $(\gamma_X \circ (c + 2))(x)(\delta) = c(x)(\delta)$, so may conclude that $y \in c(x)(\delta)$.

Then we need to show that $\forall o \in O$ we have $y \overset{o}{\not\to}$. Again using definition 4.7.1 we need to show that $(\gamma_X \circ (c+2))(y)(o) = \emptyset$. Similar to the derivation above we get that $(\gamma_X \circ (c+2))(y)(o) = c(y)(o)$. Since $y \in c(x)(\delta)$, we know that $x \xrightarrow{\delta} y$ in (X, c), and since (X, c) is anomaly free, we know that $\forall o \in O$ it holds that $y \overset{o}{\not\to}$. From definition 4.14 we can see that for output labels, γ_X does not add or change any transitions with respect to the original transition structure. We may thus conclude that indeed $(\gamma_X \circ (c+2))(y)(o) = \emptyset$.

Quiescence reducible:
Let $x, y \in X$ such that $y \xrightarrow{\delta} x$. We need to prove that $x \lesssim y$. Since we know that δ transitions are always loops, we get that $x = y$. Now we get that $x \lesssim y$ with the simulation $R = \{(x, x) \mid x \in X\}$.

Quiescence stable:
Let $x, y \in X$ such that $x \xrightarrow{\delta} y$. We need to prove that $\exists y', y'' \in X$ such that $x \xrightarrow{\delta} y'$, $y' \xrightarrow{\delta} y''$ and $y'' \lesssim y$. Since we know that δ transitions are always loops, we get that $x = y$. So, we take $y = y' = y'' = x$. This gives that $x \xrightarrow{\delta} y'$, $y' \xrightarrow{\delta} y''$. Again using the simulation $R = \{(x, x) \mid x \in X\}$ we get $y'' \lesssim y$. $\square$

Lastly we show that determinization also preserves the properties of being anomaly free, quiescence reducible and quiescence stable.

Proposition 6.11. The functor $\mathsf{det}\colon \mathsf{Coalg}(F_{\Delta,c}) \to \mathsf{Coalg}(F_{\Delta,c,d})$ preserves being anomaly free, quiescence reducible and quiescence stable.

Proof. Let (X, c) be an anomaly free, quiescence reducible, quiescence stable $F_{\Delta,c}$-coalgebra. We first write out $\mathsf{det}((X, c)) = (\mathcal{P}(X), \beta_X \circ \mathcal{P}(c))$. We now have to show that for $U \subseteq X$ the properties also hold. We write out

$$\begin{aligned}(\beta_X \circ \mathcal{P}(c))(U)(\delta) &= \beta_X(\{c(x) \mid x \in U\})(\delta) \\ &= \bigcup_{x \in U} c(x)(\delta)\end{aligned}$$

We now prove the properties separately.

Anomaly free:
Let $U, V \in \mathcal{P}(X)$ such that $U \xrightarrow{\delta} V$. Using definition 4.7.2 we know that $(\beta_X \circ \mathcal{P}(c))(U)(\delta) = V$, hence $V = \bigcup_{x \in U} c(x)(\delta)$. We have to prove that $\forall o \in O$ it holds that $V \overset{o}{\not\to}$. Because (X, c) is anomaly free, we know that for all the $y \in X$ such that $y \in \bigcup_{x \in U} c(x)(\delta)$, we have that $\forall o \in O$ it holds that $y \overset{o}{\not\to}$. From this we may conclude that indeed $\forall o \in O$ it holds that $V \overset{o}{\not\to}$.

Quiescence reducible: Let $U, V \in \mathcal{P}(X)$ such that $U \xrightarrow{\delta} V$. Using definition 4.7.2 we know that $(\beta_X \circ \mathcal{P}(c))(U)(\delta) = V$, hence $V = \bigcup_{x \in U} c(x)(\delta)$. We have to prove that $V \lesssim U$ Because (X, c) is quiescence reducible, we know that for all the $y \in V$ such that $\exists x \in U$ with $c(x)(\delta) = y$, we have that $y \lesssim x$, from this we get that also $V \lesssim U$.

Quiescence stable:
Let $U, V \in \mathcal{P}(X)$ such that $U \xrightarrow{\delta} V$. Using definition 4.7.2 we know that $(\beta_X \circ \mathcal{P}(c))(U)(\delta) = V$, hence $V = \bigcup\limits_{x \in U} c(x)(\delta)$. We have to prove that $\exists V', V'' \in \mathcal{P}(X)$ such that $U \xrightarrow{\delta} V'$, $V' \xrightarrow{\delta} V''$, and $V'' \lesssim V$. Because (X, c) is quiescence stable, we know that $\forall x, y \in U$ such that $x \xrightarrow{\delta} y$ there exist $y', y'' \in (X)$ such that $x \xrightarrow{\delta} y'$, $y' \xrightarrow{\delta} y''$ and $y'' \lesssim y$. Note that the of V is exactly the set of $y \in X$ such that $x \xrightarrow{\delta} y$. We now take $V' = \{y' \mid x \xrightarrow{\delta} y, x \in U\}$ and $V'' = \{y'' \mid y' \xrightarrow{\delta} y'', y' \in V'\}$. Then we get that $U \xrightarrow{\delta} V'$ and $V' \xrightarrow{\delta} V''$. Also $\forall y \in V$ we have that $\exists y'' \in V''$ with $y'' \lesssim y$. From this we get that also $V'' \lesssim V$. □

6.3 Suspension Coalgebras

In the previous section we have seen that the LTS constructions that we have formalized in 4.3 do indeed lead to coalgebras that are anomaly free, quiescence reducible and quiescence stable. In this section we define the subcategory of suspension coalgebras and give the final object for this category.

Definition 6.12. Let $\mathsf{Coalg}(F_{\Delta,c,d})$ be the category of $F_{\Delta,c,d}$-coalgebras. We define the category of suspension coalgebras $\mathcal{SC}$ as the subcategory of $F_{\Delta,c,d}$-coalgebras that are anomaly free, quiescence reducible and quiescence stable.

Now that we have this subcategory of $F_{\Delta,c,d}$-coalgebras, we investigate the final object for suspension coalgebras. We give the following coalgebra and then show that it is the final object in $\mathcal{SC}$.

Definition 6.13. We define the $F_{\Delta,c,d}$-coalgebra $(\mathcal{SL}, d)$ where $\mathcal{SL}$ is the set of all suspension languages and the map $d\colon \mathcal{SL} \to (I \to \mathcal{SL}) \times ((O + \{\delta\}) \rightharpoonup_{ne} \mathcal{SL})$ is defined such that for $L \in \mathcal{SL}$, $i \in I$, $o \in O + \{\delta\}$ we have

$$\begin{aligned} \pi_1(d(L))(i) &= L_i \\ \pi_2(d(L))(o) &= \begin{cases} L_o & \text{if } L_o \neq \emptyset \\ \perp & \text{otherwise} \end{cases} \end{aligned}$$

To show that the above coalgebra is indeed the final suspension coalgebra we first need to show that if we apply the map d on a suspension language and a label, we again get a suspension language. Then we need to show that for every suspension coalgebra there exists a unique mapping into $(\mathcal{SL}, d)$. For this, we use a restriction of the map traces that we already defined in 5.6. Because we have that every suspension coalgebra is an $F_{\Delta,c,d}$-coalgebra, we can use the results of the traces that we have seen in propositions 5.9 and 5.10 to show that $(\mathcal{SL}, d)$ is the final suspension coalgebra.

As said, we first show that the structure map is of the right type.

Proposition 6.14. The map $d\colon \mathcal{SL} \to (I \to \mathcal{SL}) \times ((O + \{\delta\}) \rightharpoonup_{ne} \mathcal{SL})$ is well typed.

Proof. We first point out that from 5.4 we already know that d is well typed with respect to being non-empty, prefix closed, non-blocking and input enabled. We now prove that d is also well typed with respect to we being anomaly free, quiescence reducible and quiescence stable.

Let $L \in \mathcal{SL}$. Let $a \in \Sigma^\delta$. We prove each of the required properties individually, and make a case distinction whether $a \in I$ or $a \in O^\delta$.

Anomaly free:
<u>Case $a \in I$</u>:
We have that $\pi_1(d(L))(a) = L_a$. Let $\sigma \in L_a$ and assume we can write $\sigma = \sigma_1 \delta o \sigma_2$, for some $\sigma_1, \sigma_2 \in (\Sigma^\delta)^*$ and $o \in O$. Then from the definition of derivative we get that $a\sigma_1\delta o\sigma_2 \in L$. Since L is anomaly free, this can not be the case. Hence by contradiction we get that L_a is also anomaly free.
<u>Case $a \in O^\delta$</u>:
If $\pi_2(d(L))(a) = \perp$, then d is of the right type, as the second projection of $d(L)$ is a partial map. This means that it is allowed to be undefined. If $\pi_2(d(L))(a) = L_a$, then the proof is similar to the other case.

Quiescence reducible:
<u>Case $a \in I$</u>:
We have that $\pi_1(d(L))(a) = L_a$. Let $\sigma, \rho \in (\Sigma^\delta)^*$ such that $\sigma\delta\rho \in L_a$. From the definition of derivative, we get that $a\sigma\delta\rho \in L$. Since L is quiescence reducible it follows that $a\sigma\rho \in L$. Then using the definition of derivative again, we get that $\sigma\rho \in L_a$ as wanted.
<u>Case $a \in O^\delta$</u>:
If $\pi_2(d(L))(a) = \perp$, then d is of the right type, as the second projection of $d(L)$ is a partial map. This means that it is allowed to be undefined. If $\pi_2(d(L))(a) = L_a$, then the proof is similar to the other case.

Quiescence stable:
The proof is similar to the proof for quiescence reducible. □

The next thing we show is that if there exist multiple $F_{\Delta,c,d}$-homomorphisms mapping to $(\mathcal{SL}, d)$, that this means that they are equal. To show this we use the uniqueness of the map traces that we have defined in chapter 5.

Proposition 6.15. Let $(\mathcal{SL}, d)$ be the $F_{\Delta,c,d}$-coalgebra defined above. Let (X, c) be a suspension coalgebra. Let $h, g : X \to \mathcal{SL}$ be two $F_{\Delta,c,d}$-homomorphisms. Then $h = g$.

Proof. We use the following diagram. Here the injective map $i : \mathcal{SL} \hookrightarrow \mathcal{L}_{\text{pin}}$ is the inclusion $\mathcal{SL} \subset \mathcal{L}_{\text{pin}}$.

$$\begin{tikzcd}
X \arrow[r, shift left, "h"] \arrow[r, shift right, "g"'] \arrow[rr, bend left, "\mathsf{traces}"] \arrow[d, "c"'] & \mathcal{SL} \arrow[r, hook, "i"] \arrow[d, "d"] & \mathcal{L}_{\text{pin}} \arrow[d, "d"] \\
F_{\Delta,c,d}(X) \arrow[r, shift left, "F_{\Delta,c,d}(h)"] \arrow[r, shift right, "F_{\Delta,c,d}(g)"'] \arrow[rr, bend right, "F_{\Delta,c,d}(\mathsf{traces})"'] & F_{\Delta,c,d}(\mathcal{SL}) \arrow[r, hook, "F_{\Delta,c,d}(i)"] & F_{\Delta,c,d}(\mathcal{L}_{\text{pin}})
\end{tikzcd}$$

From the uniqueness of traces we get that $i \circ h = \mathsf{traces}$ and $i \circ g = \mathsf{traces}$, hence $i \circ h = i \circ g$. Then because i is an injective map we get $h = g$. □

To make a clear distinction in the following proposition we will write $\mathsf{traces}_{\mathcal{SC}} : X \to \mathcal{SL}$ for the restriction of the traces map to suspension coalgebras. We show that the restricted map does indeed map states to suspension languages.

Proposition 6.16. Let (X, c) be a suspension coalgebra. Consider the map $\mathsf{traces}\colon X \to \mathcal{L}_{\text{pin}}$ defined in chapter 5. The map traces restricts so suspension languages.

Proof. Recall the functor $F_{\Delta,c,d}(X) = (I \to X) \times ((O + \{\delta\}) \rightharpoonup_{ne} X)$. Also recall that $\Sigma^\delta = I + O^\delta$. We already know that for any $F_{\Delta,c,d}$-coalgebra (X, c) and state $x \in X$ the trace set $\mathsf{traces}(x)$ is in $\mathcal{L}_{\text{pin}}$ from proposition 5.7. This gives that $\mathsf{traces}(x)$ is non-empty, input enabled, non-blocking and prefix closed.

We need to show that for any $x \in X$ the trace set $\mathsf{traces}(x)$ is also anomaly free, quiescence reducible and quiescence stable.

Anomaly free:
Let $x \in X$. Let $\sigma \in (\Sigma^\delta)^*$. We need to show that if $\sigma \in \mathsf{traces}(x)$ that there do not exist $\sigma_1, \sigma_2 \in (\Sigma^\delta)^*$ and $o \in O$ such that $\sigma = \sigma_1 \delta o \sigma_2$. We will show this by induction on $\sigma \in (\Sigma^\delta)^*$.

<u>Induction Base</u>:
Let $\sigma = \varepsilon$. Then for all $\sigma_1, \sigma_2 \in (\Sigma^\delta)^*$ and $o \in O$ we have that $\varepsilon \neq \sigma_1 \delta o \sigma_2$.

<u>Induction Hypothesis</u>:
Let $\sigma \in (\Sigma^\delta)^*$ such that $\sigma \in \mathsf{traces}(x)$. Then there is no way to write $\sigma = \sigma_1 \delta o \sigma_2$, for any $\sigma_1, \sigma_2 \in (\Sigma^\delta)^*$ and $o \in O$.

Induction Step:
Let $\sigma \in (\Sigma^\delta)^*$ and let $a \in \Sigma^\delta$ such that $a\sigma \in \mathsf{traces}(x)$. We need to show that there is no way to write $a\sigma = \sigma_1\delta o\sigma_2$ for any $\sigma_1\sigma_2 \in (\Sigma^\delta)^*$ and $o \in O$. Since $a\sigma \in \mathsf{traces}(x)$, we get from the definition of $\mathsf{traces}(x)$ that there exists $x' \in X$ such that $x \xrightarrow{a\sigma} x'$. From definition 4.7.2 we get that $\exists x'' \in X$ such that $c(x)(a) = x''$ and that $x'' \xrightarrow{\sigma} x'$. This gives that $\sigma \in \mathsf{traces}(x'')$, and we can use the induction hypothesis to get that we can not write $\sigma = \sigma_1'\delta o'\sigma_2'$ for any $\sigma_1', \sigma_2' \in (\Sigma^\delta)^*$ and $o' \in O$. In the case that $a \in I \cup O$, this concludes the induction step. In the case that $a = \delta$ we still have to check that it can not happen that $\sigma = o\sigma'$, for some $o \in O$ and $\sigma \in (\Sigma^\delta)^*$, as then $a\sigma = \delta o\sigma'$. This can indeed not happen, as (X, c) is anomaly free.

Quiescence Reducible:
Let $x \in X$. Let $\sigma, \rho \in (\Sigma^\delta)^*$. We need to show that if $\sigma\delta\rho \in \mathsf{traces}(x)$ that also $\sigma\rho \in \mathsf{traces}(x)$. We will show this by induction on $\sigma \in (\Sigma^\delta)^*$.

Induction Base:
Let $\sigma = \varepsilon$. Then we need to prove that if $\delta\rho \in \mathsf{traces}(x)$ we also have that $\rho \in \mathsf{traces}(x)$. Since $\delta\rho \in \mathsf{traces}(x)$ we know that $\exists y \in X$ such that $x \xrightarrow{\delta\rho} y$. From definition 4.7.2 we know that there exists $x' \in X$ such that $c(x)(\delta) = x'$ and $x' \xrightarrow{\rho} y$. From this we can see that $\rho \in \mathsf{traces}(x')$. This also gives that $x \xrightarrow{\delta} x'$. Since (X, c) is quiescence reducible, we know that $x' \lesssim x$. This, together with the fact that $\rho \in \mathsf{traces}(x')$, we can conclude that $\rho \in \mathsf{traces}(x)$ as needed.

Induction Hypothesis:
Let $\sigma, \rho \in (\Sigma^\delta)^*$, such that $\sigma\delta\rho \in \mathsf{traces}(x)$. Then $\sigma\rho \in \mathsf{traces}(x)$.

Induction Step:
Let $\sigma, \rho \in (\Sigma^\delta)^*$. Let $a \in \Sigma^\delta$ and assume $a\sigma\delta\rho \in \mathsf{traces}(x)$. We need to show that $a\sigma\rho \in \mathsf{traces}(x)$. Since $a\sigma\delta\rho \in \mathsf{traces}(x)$ we know from the definition of traces that $\exists x' \in X$ such that $x \xrightarrow{a\sigma\delta\rho} x'$. Then we get from definition 4.7.2 that $\exists x'' \in X$ such that $c(x)(a) = x''$ and that $x'' \xrightarrow{\sigma\delta\rho} x'$. This gives that $\sigma\delta\rho \in \mathsf{traces}(x'')$. We can now use the induction hypothesis to get that $\sigma\rho \in \mathsf{traces}(x'')$. From the definition of traces we thus get that $\exists x''' \in X$ such that $x'' \xrightarrow{\sigma\rho} x'''$. From this , this together with the fact that $c(x)(a) = x''$, we get from definition 4.7.2 that $x \xrightarrow{a\sigma\rho} x'''$. Using the definition of traces we can conclude that $a\sigma\rho \in \mathsf{traces}(x)$.

Quiescence Stable:
Let $x \in X$. Let $\sigma, \rho \in (\Sigma^\delta)^*$, We need to show that if $\sigma\delta\rho \in \mathsf{traces}(x)$ that also $\sigma\delta\delta\rho \in \mathsf{traces}(x)$. We will show this by induction on $\sigma \in (\Sigma^\delta)^*$.

Induction Base:
Assume $\sigma = \varepsilon$. We have to show that if $\delta\rho \in \mathsf{traces}(x)$ that then also $\delta\delta\rho \in \mathsf{traces}(x)$. Since $\delta\rho \in \mathsf{traces}(x)$ we know that $\exists y \in X$ such that $x \xrightarrow{\delta\rho} y$. From

definition 4.7.2 we know that there exists $x' \in X$ such that $c(x)(\delta) = x'$ and $x' \xrightarrow{\rho} y$. From this we can see that $\rho \in \mathsf{traces}(x')$. This also gives that $x \xrightarrow{\delta} x'$. Since (X, c) is quiescence reducible, we know that there exist x'', x''' such that $x \xrightarrow{\delta} x''$, $x'' \xrightarrow{\delta} x'''$ and $x''' \lesssim x'$. From $x''' \lesssim x'$, together with the fact that $\rho \in \mathsf{traces}(x')$, we can conclude that $\rho \in \mathsf{traces}(x''')$, Then using that $x \xrightarrow{\delta} x''$ and $x'' \xrightarrow{\delta} x'''$ we can derive that indeed $\delta\delta\rho \in \mathsf{traces}(x)$.

Induction Hypothesis:
Let $\sigma, \rho \in (\Sigma^\delta)^*$, such that $\sigma\delta\rho \in \mathsf{traces}(x)$. Then $\sigma\delta\delta\rho \in \mathsf{traces}(x)$.

Induction Step:
Let $\sigma, \rho \in (\Sigma^\delta)^*$, Let $a \in \Sigma^\delta$ and assume $a\sigma\delta\rho \in \mathsf{traces}(x)$. We need to show that $a\sigma\delta\delta\rho \in \mathsf{traces}(x)$. Since $a\sigma\delta\rho \in \mathsf{traces}(x)$ we know from the definition of traces that $\exists x' \in X$ such that $x \xrightarrow{a\sigma\delta\rho} x'$. Then we get from definition 4.7.2 that $\exists x'' \in X$ such that $c(x)(a) = x''$ and that $x'' \xrightarrow{\sigma\delta\rho} x'$. This gives that $\sigma\delta\rho \in \mathsf{traces}(x'')$. We can now use the induction hypothesis to get that $\sigma\delta\delta\rho \in \mathsf{traces}(x'')$. From the definition of traces we thus get that $\exists x''' \in X$ such that $x'' \xrightarrow{\sigma\delta\delta\rho} x'''$. From this , this together with the fact that $c(x)(a) = x''$, we get from definition 4.7.2 that $x \xrightarrow{a\sigma\delta\delta\rho} x'''$. Using the definition of traces we can conclude that $a\sigma\delta\delta\rho \in \mathsf{traces}(x)$. □

We write $\mathsf{traces}_{\mathcal{SC}} : X \to \mathcal{SL}$ for the restriction of traces to suspension languages. We now show that $\mathsf{traces}_{\mathcal{SC}}$ is also an $F_{\Delta,c,d}$-homomorphism.

Proposition 6.17. The map $\mathsf{traces}_{\mathcal{SC}} : X \to \mathcal{SL}$ is an $F_{\Delta,c,d}$-homomorphism.

Proof. We consider the following diagram

$$\begin{tikzcd}
X \arrow[r, "\mathsf{traces}_{\mathcal{SC}}"] \arrow[rr, bend left, "\mathsf{traces}"] \arrow[d, "c"'] & \mathcal{SL} \arrow[r, hook, "i"] \arrow[d, "d"] & \mathcal{L}_{\text{pin}} \arrow[d, "d"] \\
F_{\Delta,c,d}(X) \arrow[r, "F_{\Delta,c,d}(\mathsf{traces}_{\mathcal{SC}})"] \arrow[rr, bend right, "F_{\Delta,c,d}(\mathsf{traces})"'] & F_{\Delta,c,d}(\mathcal{SL}) \arrow[r, hook, "F_{\Delta,c,d}(i)"] & F_{\Delta,c,d}(\mathcal{L}_{\text{pin}})
\end{tikzcd}$$

The maps $\mathsf{traces}\colon X \to \mathcal{L}_{\text{pin}}$ and $i\colon \mathcal{SL} \to \mathcal{L}_{\text{pin}}$ are both $F_{\Delta,c,d}$-homomorphisms. We also have that $i \circ \mathsf{traces}_{\mathcal{SC}} = \mathsf{traces}$. We now use lemma 2.4 from [12], from which we may conclude that $\mathsf{traces}_{\mathcal{SC}}\colon X \to \mathcal{SL}$ is then also an $F_{\Delta,c,d}$-homomorphism. □

Now that we have seen that for any suspension coalgebra (X, c) we have that the unique $F_{\Delta,c,d}$-homomorphism $\mathsf{traces}_{\mathcal{SC}}$ maps it to $(\mathcal{SL}, d)$, we may conclude that $(\mathcal{SL}, d)$ is indeed the final object in the category of suspension coalgebras.

Corollary 6.18. The final object of $\mathcal{SC}$ is $(\mathcal{SL}, d)$.

Just as we had for $F_{\Delta,c,d}$-coalgebras, we may conclude that since the traces map is the behavior map for suspension coalgebras, we get that for suspension coalgebra, it is also the case that behavioral equivalence is equality of trace sets.

6.4 The Coalgebraic uioco Characterization

We conclude by referring back to theorem 2.24. As we have seen the uioco characterization can be obtained by applying deltafication, chaotic completion and determinization, and then considering the trace set. In the coalgebraic setting this is also the case. We give the coalgebraic uioco characterization in terms of the constructions and show that it is indeed a suspension languages.

Definition 6.19. Let (X, c) be an F-coalgebra. Let $(Y, b) = \mathsf{det}(\Xi(\Delta((X, c))))$. Then for any $y \in Y$ we define the uioco characterization as

$$\langle y \rangle_{\mathsf{uioco}} := \mathsf{traces}(y)$$

Now we can use the results we got from our coalgebraic description of the constructions to get a similar result to theorem 2.24.

Theorem 6.20. Let (X, c) be an F-coalgebra. Let $(Y, b) = \mathsf{det}(\Xi(\Delta((X, c))))$. Then for any $y \in Y$, the uioco characterization $\langle y \rangle_{\mathsf{uioco}}$ is a suspension language.

Proof. From the definitions of Δ, Ξ and det we know that (Y, b) is an $F_{\Delta,c,d}$-coalgebra. Combining this with 6.2 we know that (Y, b) is a suspension coalgebra. From 6.16 we can conclude that indeed $\forall y \in Y$ we have that $\langle y \rangle_{\mathsf{uioco}} = \mathsf{traces}(y)$ is a suspension language. $\square$

Remark 6.21. As mentioned in remark 2.27, we do not immediately get that for every suspension language L there exists a specification $\mathcal{S}$ such that the uioco characterization of $\mathcal{S}$ is equal to L. Translating this statement to coalgebra, this means that we do not immediately get that given a suspension language L there exists an F-coalgebra, such that the construction above leads to L. We can however obtain something which is quite similar. Instead of obtaining such an F-coalgebra, one can obtain a suited $F_{\Delta,c,d}$-coalgebra. This can be achieved by iteratively applying the behavior map d from the final $F_{\Delta,c,d}$-coalgebra $(\mathcal{SC}, d)$ on L to create an $F_{\Delta,c,d}$-coalgebra. This then creates a $F_{\Delta,c,d}$-coalgebra for which the states produces trace sets that are exactly equal to the language derivatives of L. We leave working out the exact details of this idea as future work.

Chapter 7

Conclusions and Future Work

In this thesis we provided a coalgebraic description of an interesting notion in the field of model based testing, namely the uioco conformance relation. With this, we tried to open up new perspectives which can be used to consider model based testing theory from a coalgebraic point of view.

We have seen a coalgebraic description of the characterization for the uioco relation that is based on LTS constructions [6, 7, 19]. To achieve this we used a framework for defining LTS constructions as functors between categories of coalgebras, using natural transformations [10, 14]. We have worked out that the constructions on LTSes can be seen as functors between categories of coalgebras with a certain type. Then, we have translated the properties that define a suspension language from a definition based on traces, to a definition based on states of an LTS. With these LTS properties we have defined suspension coalgebras. As a correctness check we showed that applying our constructions to labeled transition systems indeed leads to suspension coalgebras. Lastly we considered the final coalgebra for suspension coalgebra, from which we found that behavioral equivalence for suspension coalgebra matches equality of trace sets. In recent work [11] we have already seen the ioco relation being used as an example of the notion of a coalgebraic uncertain bisimulation. With this work we tried to open up ways to achieve similar results for the uioco relation.

For testers, the work in this thesis can be used to obtain the uioco characterization for any specification that is modeled as an LTS with inputs and output. Because we explicitly mention all the types that are used in the constructions, our work lends itself quite nicely for implementations using for example a functional languages. As future work for this thesis we would like to investigate whether we can describe a construction that can, given a specification, create a uioco conforming implementation, using the uioco characterization and the final coalgebra we have described in chapter 6.

As pointed out in remark 6.21 we did not work out whether it holds that for every suspension language L there exists a specification $\mathcal{S}$ that $\langle \mathcal{S} \rangle_{\mathsf{uioco}} = L$. This is also not worked out in [6], but it is only stated that it is not possible using the method that was used for the ioco characterization. As this thesis gives a different point of view for uioco characterizations, it could also open up different approaches for obtaining specifications from suspension languages.

Another part of our construction that can be worked out further is showing that for a suspension coalgebra the traces map produces a minimal suspension coalgebra which produces the same trace set. Alongside this, one could describe a construction for creating canonical LTSes given a suspension language. For such a canonical system we want that the trace set it produces exactly matches the suspension language provided. This could be achieved by iteratively applying the derivation map we have found when considering the final suspension coalgebra. This would then produce a suspension coalgebra from any suspension language.

The framework we have shown can be used to define constructions on different transition structures such as finite automata and Mealy machines. To achieve this one has to specify the underlying functors, and give a natural transformation of the right type that captures the transformation of the transition structure. We also think that the theory we built up, would lend itself particularly well to a formalization in a proof assistant like Coq [1]. This is because the types of the LTSes we come across are not too complicated. Also, it should be straightforward to implement the properties of being anomaly free, quiescence reducible and quiescence stable, as they are all defined using the inductive transition relation.

Bibliography

[1] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

[2] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner, editors. *Model-Based Testing of Reactive Systems, Advanced Lectures [The volume is the outcome of a research seminar that was held in Schloss Dagstuhl in January 2004]*, volume 3472 of *Lecture Notes in Computer Science*. Springer, 2005.

[3] Marie-Claude Gaudel. Testing can be formal too: 30 years later. In Bertrand Meyer, editor, *The French School of Programming*, pages 17–45. Springer, 2024.

[4] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

[5] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016.

[6] Ramon Janssen. Refinement and partiality for model-based testing. PhD Thesis, Radboud University, 2022.

[7] Ramon Janssen and Jan Tretmans. Matching implementations to specifications: the corner cases of ioco. In *SAC*, pages 2196–2205. ACM, 2019.

[8] Jurriaan Rot. Category theory and coalgebra, lecture 7: Non-deterministic systems. Raboud University, 2022.

[9] Jurriaan Rot, Marcello M. Bonsangue, and Jan Rutten. Proving language inclusion and equivalence by coinduction. *Inf. Comput.*, 246:62–76, 2016.

[10] Jurriaan Rot, Bart Jacobs, and Paul Blain Levy. Steps and traces. *J. Log. Comput.*, 31(6):1482–1525, 2021.

[11] Jurriaan Rot and Thorsten Wißmann. Bisimilar states in uncertain structures. In *CALCO*, volume 270 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[12] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. Modern Algebra.

[13] Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing the powerset construction, coalgebraically. In *FSTTCS*, volume 8 of *LIPIcs*, pages 272–283. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[14] Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Log. Methods Comput. Sci.*, 9(1), 2013.

[15] Mark Timmer, Hendrik Brinksma, and Mariëlle Ida Antoinette Stoelinga. *Model-Based Testing*, pages 1–32. Number 30 in NATO Science for Peace and Security Series D: Information and Communication Security. IOS, Netherlands, April 2011.

[16] Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Softw. Concepts Tools*, 17(3):103–120, 1996.

[17] Jan Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008.

[18] Jan Tretmans. Lecture slides: A theory of model-based testing with labeled transition systems, the uioco theory. Raboud University, 2022.

[19] Jan Tretmans and Ramon Janssen. Goodbye ioco. In *A Journey from Process Algebra via Timed Automata to Model Learning*, volume 13560 of *Lecture Notes in Computer Science*, pages 491–511. Springer, 2022.

[20] Machiel van der Bijl, Arend Rensink, and Jan Tretmans. Compositional testing with ioco. In *FATES*, volume 2931 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2003.

[21] Tim A. C. Willemse. Heuristics for ioco -based test-based modelling. In *FMICS/PDMC*, volume 4346 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2006.