Coalgebraic Modelling

Applications in Automata Theory and Modal Logic

Helle Hvid Hansen

Coalgebraic Modelling

Applications in Automata Theory and Modal Logic

Cover design by Hartmut Fitz & Helle Hvid Hansen
Printed and bound by Ipskamp Drukkers B.V., Enschede
ISBN: 978-90-8659-308-8
IPA Dissertation Series 2009-07

Nederlandse Organisatie voor Wetenschappelijk Onderzoek

The work reported in this thesis has been carried out at the Vrije Universiteit Amsterdam and the Centrum Wiskunde & Informatica (CWI), under the auspices of the research school IPA (Institute for Programming research and Algorithmics). The research was funded by the Netherlands Organization for Scientific Research (NWO) under the projects C-Quattro (612.000.316) and INFINITY (FOCUS/BRICKS grant 642.000.502).

VRIJE UNIVERSITEIT

Coalgebraic Modelling

Applications in Automata Theory and Modal Logic

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op donderdag 14 mei 2009 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Helle Hvid Hansen

geboren te Zuid-Korea

promotor:	prof.dr. J.J.M.M. Rutten
copromotoren:	dr. Y. Venema
	dr. C.A. Kupke

Acknowledgments

When I first came to Amsterdam in 1994, I was only planning to stay for a few months. I was still restless, had many plans for travelling the world, and I certainly did not think that one day I would take on a PhD position here. The reason I stayed is clear to me. My feeling of being at home in Amsterdam is due to the many great people I met over the years. Here I will try to convey my gratitude to the people who made the writing of this thesis possible, and who made the past 4 years into an enjoyable period of academic and personal growth.

First of all, I want to thank my promotor Jan Rutten for sharing his ideas with me, while at the same time giving me the freedom to choose my subjects of study. Chapters 3 and 4 of this thesis are both based on Jan's ideas. My interest in coalgebra started with reading Jan's papers. I admire his dedication to the subject and his ability to make it accessible through clear, uncomplicated writing. I also want to thank Jan for his support and encouragement, especially during the final writing stages. His careful supervision and continued support were vital to the successful completion this thesis.

I am grateful to Yde Venema for his willingness to become my copromotor, when I realised that I needed a modal logic anchor in my work. Since my time as a student at the UvA, Yde's research and teaching has always been a great source of inspiration. Any understanding I have of modal logic is thanks to him. The work presented in Chapter 5 of this thesis was initially suggested by Yde. I also want to thank Yde for his exceptional generousity of spirit; his advice and support over the years has been invaluable to me.

Clemens Kupke has been so much more than my second copromotor. I want to thank Clemens for being a great colleague, perfect office mate, dear friend and ideal day-to-day supervisor. I could always rely on him when I had questions about category theory, coalgebra and logic, and I benefited greatly from our discussions. Our cooperation was always enjoyable and enlightening—Chapter 5 is a direct product of it. I was sad to see him leave for London, and I hope that we can find the opportunity to collaborate again in the future.

I thank the members of my committee—Johan van Benthem, Marcello Bonsangue, Mai Gehrke, H. Peter Gumm, Bart Jacobs, Jan Willem Klop and Erik de Vink—for taking time out of their busy schedules to read my dissertation, and for providing numerous suggestions that helped me improve the final manuscript.

During my time as a PhD student, I had the pleasure of being part of two research groups. Officially I was employed at the Theoretical Computer Science section at the VU, but I was also part of the SEN3 group at the CWI.

I very much enjoyed my time at the VU, and I thank my VU colleagues—Rena Bakhshi, Jörg Endrullis, Wan Fokkink, Clemens Grabmayer, Ariya Isihara, Jeroen Ketema, Jan Willem Klop, Cynthia Kop, Femke van Raamsdonk, Roel de Vrijer and Paulien de Wind—for creating a warm and stimulating atmosphere. Even after living in Amsterdam for 10 years, my integration into Dutch society was brought to the next level by the lively lunch discussions which spanned topics from academic life and the news to bicycle maintenance and Jan Willem's time in the army. I especially want to thank Wan Fokkink. Although not directly involved in my supervision, Wan took an active interest in my work and well-being, and I could always turn to him for advice. Many thanks to Clemens, Jan Willem, Roel and Wan for the dinners at their homes. I have especially fond memories of our cycling trip to Roel's house on the Vecht. Thanks also to Clemens, Rena and Jörg for organising the sushi dinner. Moreover, I enjoyed sharing office with Anna Tordai, and I am thankful to Elly Lammers and Shirley Chedi for their assistance in the pre-PhD defence formalities.

The CWI is an excellent place to do research. Since I joined SEN3, the group grew substantially with a lively mix of PhD students and postdocs of numerous nationalities. Thanks to all my colleagues for making SEN3 a stimulating work environment and a considerable addition to my social agenda. Special thanks to Farhad Arbab, who was (co)projectleader of the C-Quattro project, for letting me follow my own research direction; to Dave Clarke for introducing me to the group's research during the first months, when my promotor Jan Rutten was on sabbatical; and to David Costa for helping me learn Haskell and our enjoyable cooperation. For various reasons I also want to thank Frank de Boer, Stephanie Kemper, Christian Köhler, Young-Joo Moon and Milad Niqui. Special thanks to Yanjing Wang for the delicious Chinese dinner. Very special thanks to Alexandra Silva and José Proença for the fantastic food-odyssey in Portugal. In Alexandra, I found a fellow food-lover and gym-enthousiast. I want to thank her for the many dinner parties, for teaching me about Portuguese food, and for being a great friend and inspiring colleague.

Although I was not officially associated with the ILLC during my PhD studies, with Yde Venema as my copromotor I had the privilege of regular visits there. I especially enjoyed discussions and socialising with the following people: Johan van Benthem, Nick Bezhanishvili, Balder ten Cate, Gaëlle Fontaine, Eric Pacuit, Alessandra Palmigiano, Raul Leal Rodriguez and Jacob Vosmaer.

Special thanks to Gaëlle for joining me on holidays in Sicily and the Aeolian islands, and to Nick for the visit in Leicester.

I am thankful to my coauthors—David Costa, Clemens Kupke, Eric Pacuit and Jan Rutten—for the pleasant, fruitful collaboration. Thanks are also due to Jean-Éric Pin for our discussions on subsequential transducers during his visit to the CWI in January 2006. These laid a basis for the work presented in Chapter 4.

I met many people in the coalgebra community who inspired me with their enthusiasm and knowledge of the subject. I benefited from discussions with Jiří Adámek, Neil Ghani, H. Peter Gumm, Peter Hancock, Ichiro Hasuo, Alexander Kurz, Prakash Panangaden, Dirk Pattinson, Cesar Sanchez, Lutz Schröder and Ana Sokolova. I am especially grateful to H. Peter Gumm for inviting me and Clemens Kupke to Marburg, and for his hospitality during our stay. My understanding of coalgebraic modal logic was much improved through our discussions on the topic.

Furthermore, I want to thank my new employers, Jos Baeten and Bas Luttik, for their flexibility during the preparation of this final manuscript. Thanks also to Paul van Tilburg for introducing me to ContextFreeArt, which I used to design the cover.

Many thanks to all my friends in and out of Amsterdam for providing welcome distractions. Friday nights at the Oorlam were made the perfect place to wrap up the week by the Hermeneutic Heideggers—in particular, Anna, Breanndán, Caroline, David, Fabrice, Henk, Marie, Tikitu and Will. I discovered Argentinian tango thanks to Nikolay. Special thanks to Jacobien, Lillian and Sunny for their friendship since my early days in Amsterdam, and to Freda, Mirjana and Nicole for the many visits and being just an email away.

Finally, my warmest thanks to Hartmut for sharing the ups and downs of thesis writing with me, and for being a loving partner. Last, but not least, I am grateful to my family in Denmark, especially my mother and my sister, for their love and support during all my years away from home.

Helle Hvid Hansen
Amsterdam, 15 March 2009.

Contents

Chapter 1

Introduction

1.1 Coalgebraic modelling

In theoretical computer science, state-based systems are widely used models of concrete systems such as digital hardware components, software programs and distributed systems. The purpose of modelling systems as formal structures is to be able to study and reason about them using mathematical techniques.

Coalgebra provides a category theoretical framework for studying the behaviour and properties of state-based systems. The basic concept of a coalgebra formalises the black-box view of a system, where knowledge of the system's state can only be obtained by observing the external behaviour of the system. The type of observations and transitions which can be made in such a system is specified by a functor. Formally, given a functor T on a category C, a *coalgebra for T (T-coalgebra)* is a pair $\langle X, \xi \rangle$ where X is an object in C, and $\xi \colon X \to T(X)$ is an arrow in C called the coalgebra structure. Different types of systems arise by varying the base category C and the type functor T. Due to the category theoretical setting, T-coalgebras come with a generic definition of morphism, and they allow a general theory, called *universal coalgebra* [129], in which generic notions of bisimulation and behaviour are fundamental concepts.

The definition of a coalgebra is sufficiently general to include numerous, very different types of systems. Already in the simple setting where C is the category Set of sets and functions, we find that many familiar structures can be seen as coalgebras. These include streams, trees, classical automata, Mealy and Moore type state machines, labelled transition systems, Kripke structures [129], various types of probabilistic systems [16], and topological spaces [49]. We mention that by instantiating the coalgebraic definition of a bisimulation for the examples of deterministic automata, Kripke frames and labelled transition systems we obtain the well known definitions of bisimulation from automata theory [77], modal logic [25] and process theory [109, 97], respectively. Some of these examples are presented in detail in the next chapter.

Coalgebras over base categories other than Set are useful when the desired

semantics does not fit with the generic bisimilarity notion obtained from Set-based coalgebras. For example, trace semantics for non-deterministic automata and context-free grammars can be obtained by modelling these systems as coalgebras over the category of sets and relations [60, 69], see also [61]. On the other hand, we may want the semantics to capture some structure in addition to the coalgebraic one. For example, the descriptive frames of modal logic are coalgebras over the category of Stone spaces [84], continuous probabilistic systems are coalgebras over metric spaces [154], Harsanyi type spaces are coalgebras over measurable spaces [104], and linear systems are coalgebras over vector spaces [133]. In this thesis, we will only consider coalgebras over Set.

An important aspect of coalgebraic modelling is the existence and identification of final coalgebras. A T-coalgebra $\langle Z, \zeta \rangle$ is *final*, if for any T-coalgebra $\langle X, \xi \rangle$ there exists a unique T-coalgebra morphism ϕ from $\langle X, \xi \rangle$ to $\langle Z, \zeta \rangle$. Intuitively, the elements of a final T-coalgebra represent all possible behaviours of T-coalgebras, hence a formal semantics of T-coalgebras is obtained via the final map ϕ. For example, the set of formal languages over an alphabet A is a final deterministic automaton where the final map sends a state s to the language accepted from s. The existence and uniqueness of the final map give rise to a definition principle and a proof principle. Their use is often referred to as definition and proof *by coinduction*, respectively. The *coinductive proof principle* states that if two elements in a final coalgebra are bisimilar, then they are identical. The *coinductive definition principle* refers to the use of finality to obtain a T-coalgebra morphism from a set X to Z by defining a T-coalgebra structure on X. For example, binary operations on the final T-coalgebra can be defined by taking $X = Z \times Z$.

Final coalgebras have been used to give formal semantics to processes [146], objects and classes [67], functional programs [147], and reactive programs [135]. Other, more recent, coalgebra-driven contributions in computer science are found in the areas of component-based software engineering [10, 13, 143], security [44, 63] and concurrency [62].

1.2 Modal logic and coalgebra

Modal logic has its origins in philosophy, but it has found applications in diverse areas such as linguistics, artificial intelligence and computer science [24]. Modal logics are typically interpreted over relational structures such as Kripke models and labelled transition systems. A fundamental property of this semantics is that modal formulas are invariant under bisimulations, meaning that modal formulas cannot distinguish bisimilar states. The close relationship between bisimilarity and the expressivity of modal logic is witnessed by Van Benthem's characterisation theorem [19] which tells us that every bisimulation invariant property

of Kripke models which can be defined in first order logic, is also definable in basic modal logic. Similarly, it has been shown in [71] that any bisimulation invariant property of labelled transition systems which can be defined in monadic second order logic, is also definable in the modal μ-calculus. In contrast with first order logic and monadic second order logic, modal logics often have decidable satisfiability and validity problems. Due to these appealing computational properties and their ability to express most interesting properties of processes, modal logics, in particular with temporal operators, have proved useful in the formal specification and verification of systems [12, 150, 151].

Taking the view that systems are coalgebras and system behaviour is bisimulation invariant, suggests that modal logic is the right language for reasoning about behaviours in a more general sense. Indeed, research has shown that modal logic over (labelled) transition structures can be seen as an instance of a general theory known as *coalgebraic modal logic* (cf. Pattinson [111, 112]). In coalgebraic modal logic, the truth of modalities is defined via predicate liftings. A predicate lifting λ for a functor T lifts a predicate $[\![\phi]\!]$ over X to a predicate $\lambda([\![\phi]\!])$ over $T(X)$. Given a T-coalgebra $\langle X, \xi\rangle$, the truth of the associated modality is defined by: $x \in [\![\Box\phi]\!]$ iff $\xi(x) \in \lambda([\![\phi]\!])$, for all $x \in X$.

The modal logic of Kripke models (called normal modal logic) has naturally served as an inspiration for many results in coalgebraic modal logic. In return, coalgebra has provided a larger perspective on results in normal modal logic. For example, coalgebraic modal logic comes with generic methods for proving soundness and completeness [111], algebraic duality [26, 82, 83, 88], decidability [137], expressivity [112, 136] and modular composition [39]. The view that coalgebraic modal logic is the right generalisation of modal logic is further strengthened by the results that the relationship between coalgebraic modal logic and coalgebras is formally dual to the relationship between equational logic and algebras [87], and that all modal logics axiomatised by formulas of modal depth 1 (rank 1) have a coalgebraic semantics for which they are complete [138].

Finally, we mention that although coalgebraic modal logic generalises standard modal logic in a fairly direct sense, but it is not the only approach to logics for coalgebras. Moss [102] was the first to observe that T-coalgebras give rise to a modal language with a modality ∇ whose truth is defined by lifting the satisfaction relation between states in X and formulas in Fm to a relation between $T(X)$ and $T(Fm)$. Moss' logic has the property of always being expressive, but the language is slightly non-standard and has defied a completeness proof up until very recently [85]. The finitary version of Moss' logic was extended with fixed point operators by Venema [153], and Venema & Kupke show in [86] that much of the automata theory of the modal μ-calculus can be generalised to this coalgebraic language. For the class of so-called Kripke polynomial functors, Jacobs [68] describes a many-sorted modal logic in which formulas and their sorts are inductively defined over the ingredients of the type functor T. Jacobs, build-

ing on work by Rößiger [124], proves completeness via a many-sorted canonical model construction and many-sorted Boolean algebras with operators. For a more detailed overview of these different approaches to logics for coalgebras, we refer to [152, 80].

1.3 Motivation

Our motivation for modelling systems as coalgebras is driven by mathematical curiosity as well as the desire for applications.

To start with, coalgebraic modelling can increase our understanding of a class of systems by placing them in a wider context. As it happens in mathematics, we often better understand the presence or absence of certain properties by moving to a higher level of abstraction. For example, certain properties of the structural theory of T-coalgebras can be identified in terms of preservation properties of the type functor T. One such important property is the preservation of weak pullbacks [129]. While bisimilarity always implies behavioural equivalence in T-coalgebras, the converse only holds if T preserves weak pullbacks (cf. [1, 129]). In other words, coinduction is always a sound proof principle for behavioural equivalence, however, it is only complete if T preserves weak pullbacks. A detailed study of the correspondence between (weaker) preservation properties of T and structural properties of T-coalgebras is found in [51].

From the practical point of view, an immediate benefit of a coalgebraic modelling, is that we can instantiate general results and techniques in coalgebra to the class of systems that is being modelled. In particular, we have at our hands a range of coalgebraic logical languages and derivation systems for specifying and reasoning about these systems. These logics come with general results on bisimulation invariance and expressivity, as well as uniform techniques for proving soundness and completeness. If also a final coalgebra exists, then we gain coinductive proof and definition principles.

On the other hand, coalgebra can also benefit from the study of specific examples. Results that have a natural interpretation for a certain type of systems, can lead to interesting questions at the general coalgebraic level, and to ideas that may be applied to other types of coalgebra. Furthermore, a good understanding of a particular class of systems can help develop our intuition about abstract coalgebraic notions by phrasing them in more familiar terms, for example, from automata theory or logic.

1.4 Thesis overview and contributions

We now give an introductory overview of the main chapters of this thesis, their research themes, and a brief statement of their contributions.

1.4.1 Automaton synthesis

Since the early days of automata theory, automaton synthesis has been an important field of research. Briefly stated, automaton synthesis refers to the process of constructing from a formal specification, an automaton which realises the specification. In the synthesis of deterministic automata from regular expressions (see e.g. [9, 22, 30, 106]), this means that the automaton must accept the language specified by the regular expression. More generally, in sequential synthesis (see e.g. [11, 31, 38, 117]), a specification is a stream relation given by an S1S-formula $\varphi(X, Y)$ and a realisation is a Mealy (or Moore) machine with input/output behaviour f such that for all streams σ, $\varphi(\sigma, f(\sigma))$ holds.

The techniques employed in the abovementioned work are highly diverse, and sometimes highly complex. One construction, however, is particularly interesting from the coalgebraic perspective. We are referring to Brzozowski's construction in [30] of a finite deterministic automaton from a regular expression. Brzozowski's insight was that the set *Reg* of regular expressions itself has the structure of a deterministic automaton under the operations known as 'empty word property' and Brzozowski derivative. In coalgebraic terms, Brzozowski's construction can be described as a computation of the subcoalgebra generated by e in *Reg* while working modulo ACI-equivalence (Associativity-Commutativity-Idempotence of $+$) to guarantee termination.

The coalgebraic view on Brzozowski's algorithm suggests that it can be generalised to other coalgebra types and specification languages. Such a synthesis method would require the set of specifications to carry a coalgebraic structure, the existence of generated subcoalgebras and a suitable decidable congruence on the set of specifications. We will refer to a synthesis method based on these principles as *coalgebraic synthesis*. Indeed, Rutten describes in [132] the idea of carrying out coalgebraic synthesis of Mealy machines from specifications in 2-adic arithmetic. In particular, Rutten shows that the set of 2-adic expressions carries a Mealy machine structure, and that so-called rational 2-adic expressions can be realised by a finite Mealy machine.

In Chapter 3, we complete the coalgebraic synthesis of Mealy machines from specifications in 2-adic as well as mod-2 arithmetic by giving a method for determining the equivalence of expressions. We give upper bounds on the size of a Mealy machine constructed from a rational specification, examples demonstrating that not all finite Mealy machines arise from rational 2-adic or mod-2 specifications, and a complexity analysis of the synthesis algorithm.

1.4.2 Automata as coalgebras

Some of the best known examples of systems which can be modelled as coalgebras come from the world of automata. As we have already seen, determinis-

tic automata admit a neat coalgebraic modelling which has given the existing theory of automata, languages and regular expressions an alternative mathematical perspective. Other examples are given by nondeterministic automata, Mealy/Moore machines and weighted/probabilistic automata. This still leaves several other types of automata open for coalgebraic modelling, such as various kinds of transducers, pushdown automata, and automata operating on infinite words or trees.

We will consider the following class of automata. A subsequential transducer is a type of state machine which reacts deterministically to input letters, and produces output words on transitions and terminal output words at accepting states. As a class of automata, subsequential transducers generalise both deterministic automata and Mealy machines. This combination of language recognition and transduction makes subsequential transducers useful in the areas of lexical analysis, coding theory, and more recently, in speech processing [98, 99].

The semantics of a subsequential transducer is the partial word function it computes. It has been shown in [36, 37] that every partial word function can be realised by a minimal (possibly infinite) subsequential transducer and that every subsequential transducer can be minimised via an intermediate normalisation step. These results suggest that subsequential transducers allow a coalgebraic treatment. On the other hand, due to the combination of internal states with output in the form of words (rather than letters), it is not difficult to see that equivalence of subsequential transducers cannot be fully captured by bisimilarity.

In Chapter 4, we investigate to which degree subsequential transducers and their underlying structures can be modelled as coalgebras (over Set). We show that only normalised subsequential structures can properly be regarded as coalgebras, but also that they form a reflective subcategory of all subsequential structures, and that a final (normalised) subsequential structure exists. Furthermore, for a subclass called step-by-step subsequential structures, we present an alternative coalgebraic modelling based on differentials, and we derive a new method of determining equivalence which does not go via normalisation.

1.4.3 Coalgebra and modal logic

Most functors of interest preserve weak pullbacks, including the Kripke polynomial functors that give rise to most types of automata and transition structures, but there are also interesting functors which lack this property. Such an example is given by $\mathcal{2}^{\mathcal{2}} = \mathcal{2} \circ \mathcal{2}$, the contravariant powerset functor composed with itself. $\mathcal{2}^{\mathcal{2}}$-coalgebras are of independent interest in modal logic where they are better known as neighbourhood frames.

Neighbourhood structures generalise Kripke structures, and they have become the standard semantic tool for reasoning about non-normal modal logics, i.e., modal logics in which principles such as $\Box\varphi \wedge \Box\psi \leftrightarrow \Box(\varphi \wedge \psi)$ or $\Box\top$ need

not to hold (cf. [35, 141]). However, neighbourhood semantics has received far less attention than Kripke semantics. In particular, a definition of bisimulation for neighbourhood models has so far been lacking, except for the subclass of monotonic neighbourhood models [114, 52].

Conversely, neighbourhood frames have a special role in coalgebraic modal logic, since a predicate lifting for some functor T transforms T-coalgebras into $\mathcal{2}^{\mathcal{2}}$-coalgebras by currying: $\lambda\colon \mathcal{2} \to \mathcal{2}^{T}$ corresponds to $\hat{\lambda}\colon T \to \mathcal{2}^{\mathcal{2}}$. Thus there are at least two good reasons to study neighbourhood frames. First, because they are a natural example of a class of coalgebras for a functor that does not preserve weak pullbacks, and, second, because as a class of structures they unify coalgebraic modal logic.

In Chapter 5, we apply the coalgebraic modelling of neighbourhood frames as $\mathcal{2}^{\mathcal{2}}$-coalgebras in order to investigate the notion of state equivalence in neighbourhood structures. Apart from behavioural equivalence and $\mathcal{2}^{\mathcal{2}}$-bisimilarity, which differ since $\mathcal{2}^{\mathcal{2}}$ does not preserve weak pullbacks, we study a third equivalence notion, whose witnessing relations we call precocongruences since they are closely related to the notion of a precongruence from [1]. We show that precocongruences are a better approximation of behavioural equivalence than $\mathcal{2}^{\mathcal{2}}$-bisimilarity while still enjoying a back-and-forth style characterisation. In the second part of Chapter 5, we prove model-theoretic results for classical modal logic which generalise known results from the theory of normal modal logic. These include a Hennessy-Milner theorem for image-finite neighbourhood models, a Van Benthem style characterisation theorem, and Craig interpolation.

Contribution summary

Briefly summarised, the main contributions are:

(i) A synthesis method for constructing Mealy machines from specifications in binary arithmetic, and results that relate Mealy machines and rational 2-adic and mod-2 specifications in terms of complexity and expressivity.

(ii) A coalgebraic perspective on subsequential transducers which sheds light on existing results, but also shows that subsequential transducers are on the edge of the range of systems that are coalgebras over Set. A systematic classification of reflective subcategories of subsequential structures, and an alternative coalgebraic modelling for so-called step-by-step structures.

(iii) Model-theoretic results for classical modal logic. Firstly, a detailed study of state equivalence, including the introduction of a new general one which improves on bisimulations when the functor does not preserve weak pullbacks. Secondly, the development of model-theoretic machinery which eventually leads to a Van Benthem style characterisation theorem and Craig interpolation.

1.5 Origin of the material

Parts of the material presented in this thesis have been published previously, or are currently awaiting publication. Chapter 3 extends joint work with David Costa and Jan Rutten in [55]. Chapter 4 extends the work presented in [53]. Chapter 5 is based on joint work with Clemens Kupke and Eric Pacuit in [57], of which an extended version will soon appear as the journal article [58]. The table below summarises the relationship between chapters and publications.

Ch. 3:	[55]	H.H. Hansen, D. Costa and J.J.M.M. Rutten. Synthesis of Mealy machines using derivatives. *Proceedings CMCS 2006*. ENTCS 164, pp. 27–45.
Ch. 4:	[53]	H.H. Hansen. Coalgebraising subsequential transducers. *Proceedings CMCS 2008*. ENTCS 203, pp. 109–129.
Ch. 5:	[57]	H.H. Hansen, C. Kupke and E. Pacuit. Bisimulation for neighbourhood structures. *Proceedings CALCO 2007*. LNCS 4624, pp. 279–293.
	[58]	H.H. Hansen, C. Kupke and E. Pacuit. Neighbourhood structures: bisimilarity and basic model theory. To appear in *Logical Methods in Computer Science*.

Chapter 2

Coalgebra preliminaries

In this chapter we provide most of the basic definitions and results needed for the reading of this thesis. For a more thorough introduction to universal coalgebra, we refer to [129]. At the end of the chapter, we summarise the coalgebraic modelling of streams, deterministic automata and Kripke frames.

2.1 Sets, functors, categories

We assume the reader is familiar with basic category theoretical concepts such as category, subcategory, functor, Hom-functor, natural transformation and isomorphism in a category. The reader may consult any standard textbook on category theory, e.g. [3, 91], for these definitions and for more information on the notions defined defined below.

For an object X in a category C, we denote the *identity* C*-morphism* on X by id_X. We write $X \cong Y$, if X and Y are *isomorphic* in C. Let C and D be categories. Recall that C is a *full subcategory* of D if C is a subcategory of D, and for all objects X, Y in C, $f\colon X \to Y$ is a morphism in C iff $f\colon X \to Y$ is a morphism in D. We write $\mathsf{C} \sqsubseteq \mathsf{D}$ if C is isomorphic to a full subcategory of D.

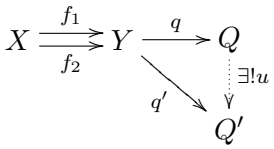
We denote by Set the category of sets and functions. A Set*-functor*, is a functor on Set, i.e., a functor $F\colon \mathsf{Set} \to \mathsf{Set}$. If F and G are Set-functors, we say that F is a *subfunctor* of G, written $F \hookrightarrow G$, if there is a natural transformation $\eta\colon F \to G$ that is injective in all components. We will often use placeholder notation for functors, for example, $A \times (-)$ denotes the functor $T(X) = A \times X$.

Two functors are of particular importance. The *covariant powerset functor* $\mathcal{P}\colon \mathsf{Set} \to \mathsf{Set}$ maps a set X to its set of subsets $\mathcal{P}(X)$, and a function $f\colon X \to Y$ to the direct image function $f[\_]\colon \mathcal{P}(X) \to \mathcal{P}(Y)$ defined by $f[C] = \{f(x) \mid x \in C\}$ for all $C \subseteq X$. The *contravariant powerset functor* $2\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{Set}$ maps a set X to $\mathcal{P}(X)$, and a function $f\colon X \to Y$ (which in $\mathsf{Set}^{\mathrm{op}}$ is an arrow $Y \to X$) to the inverse image function $f^{-1}[\_]\colon \mathcal{P}(Y) \to \mathcal{P}(X)$ defined by $f^{-1}[D] = \{x \in X \mid f(x) \in D\}$ for all $D \subseteq Y$. The *Hom*-functors in Set will usually be written

in the form of exponentiation. For sets X and Y, $Y^X = \{f\colon X \to Y\}$ is the set of functions from X to Y. For a fixed set A, the *covariant Hom-functor* $Hom(A,-)\colon \mathsf{Set} \to \mathsf{Set}$ maps a set X to X^A, and for a function $f\colon X \to Y$, $Hom(A,f) = f^A$ maps $g \in X^A$ to $f^A(g) = f \circ g$. For a fixed set B, the *contravariant Hom-functor* $Hom(-,B)\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{Set}$ maps a set X to B^X, and for a function $f\colon X \to Y$, $Hom(f,B) = B^f\colon B^Y \to B^X$ maps $g \in B^Y$ to $B^f(g) = g \circ f$.

We will make frequent use of coequalisers, coproducts, and their defining universal property. We therefore now give the general definition.

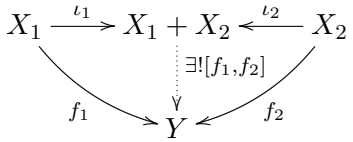
2.1.1. Definition. Let C be a category and let $f_1, f_2\colon X \to Y$ be a pair of parallel morphisms in C. A *coequaliser* of f_1 and f_2 in C is a morphism $q\colon Y \to Q$ in C such that $q \circ f_1 = q \circ f_2$, and for any $q'\colon Y \to Q'$ such that $q' \circ f_1 = q' \circ f_2$ there exists a unique morphism $u\colon Q \to Q'$ such that $q' = u \circ q$, as illustrated in the following commuting diagram.



◁

In Set, a coequaliser of two functions $f_1, f_2\colon X \to Y$ is obtained by letting E be the least equivalence relation on Y that contains $\{\langle f(x), g(x)\rangle \mid x \in X\}$, and taking $q\colon Y \to Y/E$ to be the natural quotient map, where Y/E denotes the set of E-equivalence classes on Y. In particular, if R is a relation on X and R^e is the equivalence relation generated by R on X, then the quotient map $q\colon X \to X/R^e$ is a coequaliser of the projection maps $\pi_1, \pi_2\colon R \to X$. We will also refer to a coequaliser of $\pi_1, \pi_2\colon R \to X$ simply as a coequaliser of R. Like all universal objects, coequalisers are unique up to isomorphism. We will therefore speak of *the* coequaliser of R, and often denote it by $\varepsilon_R\colon X \to X/R^e$.

2.1.2. Definition. Let C be a category. The *coproduct* of two objects X_1 and X_2 in C is an object $X_1 + X_2$ in C together with morphisms $\iota_i\colon X_i \to X_1 + X_2$, $i \in \{1,2\}$, in C with the following universal property. If $f_1\colon X_1 \to Y$ and $f_2\colon X_2 \to Y$ are morphisms in C, then there is a unique morphism $[f_1, f_2]\colon X_1 + X_2 \to Y$ such that the following diagram commutes:



◁

Products are defined dually to coproducts. We remind the reader of the definition of products and coproducts in Set. Let X and Y be sets. The product of X and Y is the span

$$X \xleftarrow{\pi_X} X \times Y \xrightarrow{\pi_Y} Y$$

where $X \times Y$ is the cartesian product of X and Y and $\pi_X \colon X \times Y \to X$, $\pi_Y \colon X \times Y \to Y$ are the projections. The coproduct of X and Y is the cospan

$$X \xrightarrow{\iota_X} X + Y \xleftarrow{\iota_Y} Y$$

where $X + Y$ is the disjoint union of X and Y, and $\iota_X \colon X \to X + Y$, $\iota_Y \colon Y \to X + Y$ are the canonical inclusions. Given two functions $f_1 \colon X_1 \to Y_1$ and $f_2 \colon X_2 \to Y_2$, their product $f_1 \times f_2 \colon X_1 \times X_2 \to Y_1 \times Y_2$ and coproduct $f_1 + f_2 \colon X_1 + X_2 \to Y_1 + Y_2$ are defined as follows:

$$\begin{array}{ll} \text{for all } x \in X_1 + X_2 : & (f_1 + f_2)(x) \;=\; f_i(x) \text{ iff } x \in X_i, i \in \{1,2\}, \\ \text{for all } \langle x_1, x_2 \rangle \in X_1 \times X_2 : & (f_1 \times f_2)(\langle x_1, x_2 \rangle) \;=\; \langle f_1(x_1), f_2(x_2) \rangle. \end{array}$$

Coproducts and products can thus be lifted to Set-functors as follows. Let F and G be Set-functors. Their product $F \times G$ and coproduct $F + G$ is defined as follows, for all sets X and all functions $f \colon X \to Y$:

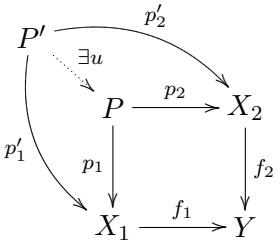
$$\begin{array}{rclrcl} (F \times G)(X) & = & F(X) \times G(X), & (F \times G)(f) & = & F(f) \times G(f), \\ (F + G)(X) & = & F(X) + G(X), & (F + G)(f) & = & F(f) + G(f). \end{array}$$

The class of *polynomial functors* is the least class of Set-functors which contains the identity Set-functor, all constant Set-functors, and is closed under products, coproducts and exponentiation with constant sets. For example, the functor $2 \times (-)^A$, whose coalgebras are deterministic automata, is polynomial. We will use a number of well known properties (to be defined below) of polynomial functors: If T is polynomial, then T preserves weak pullbacks, a final T-coalgebra exists and generated subcoalgebras exist (cf. [129, 69]).

A functor property which plays an important role in the theory of coalgebras is the preservation of weak pullbacks.

2.1.3. Definition. Let C be a category and let $f_1 \colon X_1 \to Y$ and $f_2 \colon X_2 \to Y$ be morphisms in C. A *weak pullback* of f_1 and f_2 in C is a triple $\langle P, p_1, p_1 \rangle$ where P is an object and $p_1 \colon P \to X_1$, $p_2 \colon P \to X_2$ are morphisms in C such that $f_1 \circ p_1 = f_2 \circ p_2$. Moreover, if P', $p'_1 \colon P' \to X_1$ and $p'_2 \colon P' \to X_2$ are such that $f_1 \circ p'_1 = f_2 \circ p'_2$, then there exists a morphism $u \colon P' \to P$ in C such that $p'_1 = p_1 \circ u$ and $p'_2 = p_2 \circ u$, as illustrated below. If the morphism u is unique,

then $\langle P, p_1, p_2 \rangle$ is a *pullback*.



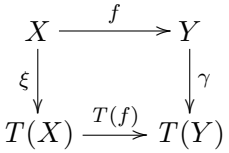
A functor $T\colon \mathsf{C} \to \mathsf{C}$ *preserves weak pullbacks* if $\langle T(P), T(p_1), T(p_2) \rangle$ is a weak pullback of $T(f_1)$ and $T(f_2)$ whenever $\langle P, p_1, p_2 \rangle$ is a weak pullback of f_1 and f_2. ◁

In Set, pullbacks can be obtained as follows. Let $f_1\colon X_1 \to Y$ and $f_2\colon X_2 \to Y$ be functions. The *pullback of* f_1 *and* f_2 in Set is the triple $(\mathrm{pb}(f_1, f_2), \pi_1, \pi_2)$, where $\mathrm{pb}(f_1, f_2) := \{\langle s_1, s_2 \rangle \in X_1 \times X_2 \mid f_1(s_1) = f_2(s_2)\}$; and $\pi_1\colon \mathrm{pb}(f_1, f_2) \to X_1$ and $\pi_2\colon \mathrm{pb}(f_1, f_2) \to X_2$ are the projections. Note that for a function $f\colon X \to Y$, the pullback $\mathrm{pb}(f, f)$ is the kernel of f, $\ker(f) = \{\langle x, x' \rangle \mid f(x) = f(x')\}$. More information about pullbacks in Set and their relevance for coalgebra can be found in [49, 51]. We note that polynomial functors and the powerset functor $\mathcal{P}$ all preserve weak pullbacks.

2.2 Coalgebras over sets

Throughout this section, we let $T\colon \mathsf{Set} \to \mathsf{Set}$ denote an arbitrary Set-functor. Coalgebras are defined over an arbitrary base category, however, in this thesis we will only consider coalgebras over Set which leads to the following definition (the general definition can be obtained in the obvious way).

2.2.1. Definition. A T*-coalgebra* is a pair $\langle X, \xi \rangle$ where X is a set and $\xi\colon X \to T(X)$ is a function. The set X is called the carrier or the state space, and ξ is called the coalgebra structure map. If $\langle X, \xi \rangle$ and $\langle Y, \gamma \rangle$ are T-coalgebras, then a T*-coalgebra morphism* from $\langle X, \xi \rangle$ to $\langle Y, \gamma \rangle$ is a function $f\colon X \to Y$ such that $\gamma \circ f = T(f) \circ \xi$, i.e., the diagram below commutes. In this case we also write: $f\colon \langle X, \xi \rangle \to \langle Y, \gamma \rangle$.



T-coalgebras and T-coalgebra morphisms form a category which we denote by $\mathsf{Coalg}(T)$. ◁

2.2.2. Remark. All notions pertaining to T-coalgebras are parametric in the functor T, but if T is clear from the context or immaterial, then we may leave it out, and simply speak of coalgebras, coalgebra morphisms etc. ◁

We sometimes want to consider a coalgebra together with a distinguished point. For example, a deterministic automaton is a coalgebra with a distinguished state, called the initial state. More generally, a *pointed T-coalgebra* $\langle X, \xi, x\rangle$ consists of a T-coalgebra $\langle X, \xi\rangle$ and a state $x \in X$. A *morphism of pointed F-coalgebras* from $\langle X, \xi, x\rangle$ to $\langle Y, \gamma, y\rangle$ is a T-coalgebra morphism $f\colon \langle X, \xi\rangle \to \langle Y, \gamma\rangle$ for which $f(x) = y$. Pointed T-coalgebras and their morphisms form a category $\mathsf{PtCoalg}(T)$.

We list a couple of useful facts. If F is a subfunctor of T, then $\mathsf{Coalg}(F)$ is isomorphic to a full subcategory of $\mathsf{Coalg}(T)$, i.e., $F \hookrightarrow T$ implies $\mathsf{Coalg}(F) \sqsubseteq \mathsf{Coalg}(T)$. Similarly, $F \hookrightarrow T$ implies $\mathsf{PtCoalg}(F) \sqsubseteq \mathsf{PtCoalg}(T)$. In $\mathsf{Coalg}(T)$, a function f is an isomorphism iff f is a bijective T-coalgebra morphism. Although we will not make much use of the following facts, we mention that surjective T-coalgebra morphisms are exactly the epimorphisms of $\mathsf{Coalg}(T)$, and injective T-coalgebra morphisms are mono in $\mathsf{Coalg}(T)$, however, monomorphisms in $\mathsf{Coalg}(T)$ are not necessarily injective, unless T preserves weak pullbacks (cf. [129, Prop. 4.7]).

We will need some basic constructions in the category $\mathsf{Coalg}(T)$.

2.2.3. Definition. Let $\langle X, \xi\rangle$ and $\langle Y, \gamma\rangle$ be T-coalgebras. A *subcoalgebra* of a T-coalgebra $\langle Y, \gamma\rangle$ is a T-coalgebra $\langle X, \xi\rangle$ such that $X \subseteq Y$ and the inclusion map $\iota\colon X \to Y$ is a T-coalgebra morphism from $\langle X, \xi\rangle$ to $\langle Y, \gamma\rangle$. ◁

Subcoalgebras are determined by their carrier set (cf. [129, Prop. 6.1]), meaning that if $\langle X, \xi\rangle$ and $\langle X, \xi'\rangle$ are subcoalgebras of $\langle Y, \gamma\rangle$, then $\xi = \xi'$, so we can simply speak of subsets $X \subseteq Y$ as subcoalgebras of $\langle Y, \gamma\rangle$. Given a state y in $\langle Y, \gamma\rangle$, if the intersection of all subcoalgebras containing y is again a subcoalgebra of $\langle Y, \gamma\rangle$, we denote this least subcoalgebra by $\langle y\rangle_Y$ and call it the *subcoalgebra generated by y in $\langle Y, \gamma\rangle$*. If T is a polynomial functor, then generated subcoalgebras always exist in $\mathsf{Coalg}(T)$, and in this case $\langle y\rangle_Y$ can be obtained essentially by taking the transition closure of $\{y\}$ in $\langle Y, \gamma\rangle$. Furthermore, we mention that, in general, if $f\colon \langle X, \xi\rangle \to \langle Y, \gamma\rangle$ is a T-coalgebra morpishm, then the image $f[X]$ of f is a subcoalgebra of $\langle Y, \gamma\rangle$. In other words, $\mathsf{Coalg}(F)$ *has image factorisation*: every T-coalgebra morphism $f\colon \langle X, \xi\rangle \to \langle Y, \gamma\rangle$ factors as a composition $i \circ e$ where $e\colon \langle X, \xi\rangle \twoheadrightarrow \langle f[X], \xi\restriction_{f[X]}\rangle$ is an epimorphism, and $i\colon \langle f[X], \xi\restriction_{f[X]}\rangle \to \langle Y, \gamma\rangle$ is a monomorphism.

2.2.4. Definition. Given T-coalgebras $\langle X, \xi\rangle$ and $\langle Y, \gamma\rangle$, $\langle Y, \gamma\rangle$ is a *quotient* of $\langle X, \xi\rangle$ if there exists a surjective T-coalgebra morphism $f\colon \langle X, \xi\rangle \twoheadrightarrow \langle Y, \gamma\rangle$. An equivalence relation $R \subseteq X \times X$ is a *congruence on* $\langle X, \xi\rangle$, if there exists

a T-coalgebra structure $\lambda\colon X/R \to T(X/R)$ such that $\varepsilon_R\colon X \to X/R$ is a T-coalgebra morphism from $\langle X, \xi\rangle$ to $\langle X/R, \lambda\rangle$ as illustrated by the commuting diagram:

$$\begin{array}{ccccc} R & \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} & X & \xrightarrow{\varepsilon_R} & X/R \\ & & \downarrow{\scriptstyle \xi} & & \vdots{\scriptstyle \lambda} \\ & & T(X) & \xrightarrow{T(\varepsilon_R)} & T(X/R) \end{array}$$

◁

Recall that if R is an equivalence relation, then the natural quotient map $\varepsilon_R\colon X \to X/R$ is the coequaliser of R. Using the fact that $\mathsf{Coalg}(T)$ has image factorisation, R is a congruence on $\langle X, \xi\rangle$ iff R is the kernel of some T-coalgebra morphism $f\colon \langle X, \xi\rangle \to \langle Y, \gamma\rangle$, i.e., $R = \ker(f) = \{\langle x, x'\rangle \mid f(x) = f(x)\}$. Congruences are conveniently characterised by the following lemma.

2.2.5. Lemma. *Let $\langle X, \xi\rangle$ be a T-coalgebra, R an equivalence relation on X, and $\varepsilon_R\colon X \to X/R$ the natural quotient map. R is a congruence on $\langle X, \xi\rangle$ iff for all $\langle x, y\rangle \in R$: $(T(\varepsilon_R) \circ \xi)(x) = (T(\varepsilon_R) \circ \xi)(y)$.*

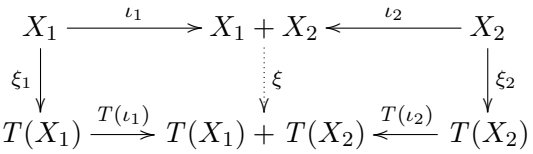
Proof. First, if R is a congruence, then there is a $\lambda\colon X/R \to T(X/R)$ such that ε_R is a T-coalgebra morphism. Hence for all $\langle x, y\rangle \in R$: $\lambda(\varepsilon_R(x)) = \lambda(\varepsilon_R(y))$, and since ε_R is a T-coalgebra morphism, $(T(\varepsilon_R) \circ \xi)(x) = (T(\varepsilon_R) \circ \xi)(y)$. Conversely, if $(T(\varepsilon_R) \circ \xi)(x) = (T(\varepsilon_R) \circ \xi)(y)$ for all $\langle x, y\rangle \in R$, then $T(\varepsilon_R) \circ \xi \circ \pi_1 = T(\varepsilon_R) \circ \xi \circ \pi_2$, and by the universal property of the coequaliser ε_R, there is a unique $\lambda\colon X/R \to T(X/R)$ such that $T(\varepsilon_R) \circ \xi = \lambda \circ \varepsilon_R$, i.e., ε_R is a T-coalgebra morphism. QED

The above lemma shows that if R is a congruence, then the induced coalgebra map $\lambda\colon X/R \to T(X/R)$ is unique. We refer to $\langle X/R, \lambda\rangle$ as *the quotient of* $\langle X, \xi\rangle$ *with* R, and denote it by $\langle X, \xi\rangle/R$. Congruences on a coalgebra $\langle X, \xi\rangle$ form a lattice (cf. [51]), and in particular, a largest congruence on $\langle X, \xi\rangle$ exists. A T-coalgebra $\langle X, \xi\rangle$ is *minimal* if $\langle X, \xi\rangle$ has no proper quotients, which means that if $f\colon \langle X, \xi\rangle \twoheadrightarrow \langle Y, \gamma\rangle$ is a surjective T-coalgebra morphism, then f must be an isomorphism. Hence $\langle X, \xi\rangle$ minimal iff the largest congruence on $\langle X, \xi\rangle$ is the identity relation.

Coproducts in $\mathsf{Coalg}(T)$ can be constructed essentially in the same way as in Set.

2.2.6. Definition. The coproduct of $\langle X_1, \xi_2\rangle$ and $\langle X_2, \xi_2\rangle$ in $\mathsf{Coalg}(T)$ is the T-coalgebra $\langle X_1 + X_2, \xi\rangle$ where $X_1 + X_2$ is the disjoint union of X_1 and X_2, and ξ is the unique function $\xi\colon X_1 + X_2 \to T(X_1 + X_2)$ obtained by the universal property of the coproduct $X_1 + X_2$ in Set, as illustrated by the following

commuting diagram.



In other words, ξ is the unique T-coalgebra structure on $X_1 + X_2$ such that the natural inclusions $\iota_1 \colon X_1 \to X_1 + X_2$ and $\iota_2 \colon X_2 \to X_1 + X_2$ are T-coalgebra morphisms. ◁

The above construction of coproducts in $\mathsf{Coalg}(T)$ applies more generally to all colimits in $\mathsf{Coalg}(T)$, since the forgetful functor $U \colon \mathsf{Coalg}(T) \to \mathsf{Set}$ creates colimits. This means that any colimit in $\mathsf{Coalg}(T)$ can be obtained by first constructing it in Set, and then supplying it in a unique way with coalgebraic structure. We refer to [129, Sec. 4.4] for more details.

2.2.7. Definition. Let $T \colon \mathsf{Set} \to \mathsf{Set}$ be a functor. A *final T-coalgebra* is a final object $\langle Z, \zeta \rangle$ in the category $\mathsf{Coalg}(T)$, which means that for any T-coalgebra $\langle X, \xi \rangle$ there exists a unique T-coalgebra morphism $f \colon \langle X, \xi \rangle \to \langle Z, \zeta \rangle$ called the *final map*. ◁

We can think of the states in a final T-coalgebra as representatives of the possible behaviours of T-coalgebras, and we are often interested in finding an explicit representation. The final map can therefore be seen as a semantics which assigns behaviour to T-coalgebra states. Hence two states are identified by the final map if and only if they have the same behaviour.

A final T-coalgebra may not exist. This is due to cardinality issues in Set, and the fact that the coalgebra structure map of a final coalgebra must be an isomorphism (cf. Lambek [90]). One class of functors for which a final coalgebra always exists is the class of polynomial functors. We refer to [1, 14, 160] for results on the existence and construction of final coalgebras.

Final coalgebras are minimal and unique up to isomorphism. Consequently, if $\langle Z, \zeta \rangle$ is a final T-coalgebra, then any minimal T-coalgebra $\langle X, \xi \rangle$ is isomorphic to the subcoalgebra $f[X]$ of $\langle Z, \zeta \rangle$, where $f[X]$ is the image of $\langle X, \xi \rangle$ under the final map $f \colon \langle X, \xi \rangle \to \langle Z, \zeta \rangle$.

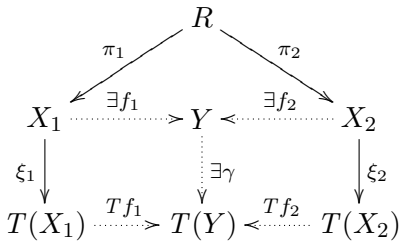
The existence of a final T-coalgebra $\langle Z, \zeta \rangle$ gives rise to a definition principle and a proof principle (cf. [129]). The *coinductive definition principle* refers to a the use of finality to obtain a map $f \colon X \to Z$ by equipping X with a T-coalgebra structure $\xi \colon X \to T(X)$. The *coinductive proof principle* refers to the use of finality to prove that two maps $f_1, f_2 \colon X \to Z$ are equal by equipping X with a T-coalgebra structure $\xi \colon X \to T(X)$ such that f_1 and f_2 are T-coalgebra morphisms. The coinductive proof principle is usually formulated in

terms of bisimulations: if z and z' are bisimilar states in $\langle Z, \zeta \rangle$, then $z = z'$ (see Theorem 2.3.6 below). After we define bisimulations in Section 2.3, it will be easy to see that the two formulations are equivalent.

2.3 Equivalence notions

T-coalgebras come with two equivalence notions which we call behavioural equivalence and bisimilarity. Both notions are essentially based on the fact that coalgebra morphisms preserve and reflect behaviour. For example, suppose that $f_1 \colon \langle X_1, \xi_1 \rangle \to \langle Y, \gamma \rangle$ and $f_2 \colon \langle X_2, \xi_2 \rangle \to \langle Y, \gamma \rangle$ are T-coalgebra morphisms and $f_1(x_1) = f_2(x_2)$. Then x_1 and x_2 must have the same behaviour. This is the idea behind *behavioural equivalence* (cf. [87]).

2.3.1. Definition. Let $\langle X_1, \xi_1 \rangle$ and $\langle X_2, \xi_2 \rangle$ be T-coalgebras. Two states $x_1 \in X_1$ and $x_2 \in X_2$ are *behaviourally equivalent* (notation: $x_1 \sim_b x_2$), if there exists a T-coalgebra $\langle Y, \gamma \rangle$ and T-coalgebra morphisms $f_i \colon \langle X_i, \xi_i \rangle \to \langle Y, \gamma \rangle$ for $i = 1, 2$ such that $\langle x_1, x_2 \rangle \in R := \mathrm{pb}(f_1, f_2)$, as illustrated here:



The cospan $\langle \langle Y, \gamma \rangle, f_1, f_2 \rangle$ in $\mathsf{Coalg}(T)$ is called a *cocongruence* between $\langle X_1, \xi_1 \rangle$ and $\langle X_2, \xi_2 \rangle$. If $\langle \langle Y, \gamma \rangle, f_1, f_2 \rangle$ is a cocongruence, then we will also refer to $R = \mathrm{pb}(f_1, f_2)$ as a cocongruence between $\langle X_1, \xi_1 \rangle$ and $\langle X_2, \xi_2 \rangle$. The relation $\sim_b$ is called *behavioural equivalence*. We call a relation R a cocongruence *on* $\langle X, \xi \rangle$ if R is a cocongruence between $\langle X, \xi \rangle$ and itself. ◁

Note that $R \subseteq X \times X$ is a congruence on $\langle X, \xi \rangle$ iff R is a cocongruence on $\langle X, \xi \rangle$ and R is an equivalence relation.

2.3.2. Remark. In [87], Kurz refers to (the kernel of) an epimorphism as a behavioural equivalence. We have chosen to use the word congruence for kernels, and reserve behavioural equivalence to denote the equivalence notion associated with congruences and cocongruences. ◁

Behavioural equivalence on a coalgebra is always the largest congruence, as shown in the following lemma (which was proved in [51, Lemma 5.10]).

2.3.3. LEMMA. *Let $\langle X, \xi\rangle$ be a T-coalgebra. If $R \subseteq X \times X$ is a cocongruence on $\langle X, \xi\rangle$, then R is contained in a congruence on $\langle X, \xi\rangle$. Consequently, behavioural equivalence is the largest congruence on $\langle X, \xi\rangle$.*

PROOF. Assume $R = \mathrm{pb}(f_1, f_2)$ is the pulback (in Set) of the T-coalgebra morphisms $f_1, f_2 \colon \langle X, \xi\rangle \to \langle Y, \gamma\rangle$. Let $e \colon \langle Y, \gamma\rangle \to \langle Y', \gamma'\rangle$ be the coequaliser of f_1 and f_2 in $\mathsf{Coalg}(T)$. Then $e \circ f_1 = e \circ f_2$ and e is a T-coalgebra morphism. It follows that $f := e \circ f_1 = e \circ f_2$ is a T-coalgebra morphism such that $R \subseteq \mathrm{pb}(f, f) = \ker(f)$. Thus R is contained in the congruence $\ker(f)$. The final statement follows from the fact that a congruence is also a cocongruence. QED

The coalgebraic notion of a bisimulation is obtained by reversing the arrows in the definition of a cocongruence. Formally, T-bisimulations are defined as follows (cf. [1]).

2.3.4. DEFINITION. Let $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ be T-coalgebras. A relation $R \subseteq X_1 \times X_2$ is a *T-bisimulation* between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ if there exists a function $\rho \colon R \to T(R)$ such that the projections $\pi_i \colon R \to X_i$ are T-coalgebra morphisms from $\langle R, \rho\rangle$ to $\langle X_i, \xi\rangle$, $i \in \{1, 2\}$.

$$\begin{array}{ccccc} X_1 & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & X_2 \\ {\scriptstyle \xi_1}\downarrow & & \vdots\,{\scriptstyle \exists \rho} & & \downarrow{\scriptstyle \xi_2} \\ T(X_1) & \xleftarrow{T(\pi_1)} & T(R) & \xrightarrow{T(\pi_2)} & T(X_2) \end{array}$$

Two states s_1 and s_2 are *T-bisimilar* (notation: $s_1 \sim s_2$), if they are linked by some T-bisimulation. The relation $\sim$ is called *T-bisimilarity*. If R is a T-bisimulation between $\langle X, \xi\rangle$ and $\langle X, \xi\rangle$, then we say that R is a T-bisimulation on $\langle X, \xi\rangle$. ◁

2.3.5. REMARK. Note that in Definition 2.3.4, we do not lose any generality by requiring that R is relation rather than an arbitrary set. For suppose $f_1 \colon \langle Y, \gamma\rangle \to \langle X_1, \xi_1\rangle$ and $f_2 \colon \langle Y, \gamma\rangle \to \langle X_2, \xi_2\rangle$ are T-coalgebra morphisms. Then the relation $R = \{\langle f_1(y), f_2(y)\rangle \mid y \in Y\}$ is a T-bisimulation: as the T-coalgebra structure map on R we can take $T(\langle f_1, f_2\rangle) \circ \gamma \circ g$, where $g \colon R \to Y$ is a right inverse of $\langle f_1, f_2\rangle \colon Y \twoheadrightarrow R$. It is straightforward to check that the projections $\pi_1 \colon R \to X_1$ and $\pi_2 \colon R \to X_2$ are T-coalgebra morphisms using the fact that f_i is a T-coalgebra morphism and $f_i = \pi_i \circ \langle f_1, f_2\rangle$ for $i \in \{1, 2\}$. ◁

We summarise a number of well known facts on bismulations. A union of T-bisimulations is again a T-bisimulation, and hence $\sim$ is the largest T-bisimulation between two T-coalgebras (cf. [129]). For any functor T, T-bisimilarity implies behavioural equivalence (this fact will also follow from Proposition 5.3.8 in Chapter 5). However, the converse only holds if T preserves weak

pullbacks. In particular, if T does not preserve weak pullbacks, then the largest congruence on T-coalgebra is not necessarily a T-bisimulation. In this case, behavioural equivalence is generally preferred over bisimilarity.

The main use of bismulations is their application as a coinductive proof principle.

2.3.6. THEOREM. *If $\langle Z, \zeta \rangle$ is a final T-coalgebra and $z, y \in Z$, then $z \sim y$ implies $z = y$.*

PROOF. Suppose $\langle z, y \rangle \in R$ for some T-bisimulation R on $\langle Z, \zeta \rangle$, i.e. there exists a T-coalgebra structure $\rho\colon R \to T(R)$ such that the projections $\pi_1, \pi_2\colon R \to Z$ are T-coalgebra morphisms. By the uniqueness of the final map, $\pi_1 = \pi_2$, hence $z = y$. QED

Coinduction is thus a way of proving that two states in a final coalgebra are identical by displaying a bisimulation linking them. Coinduction can thus be used to prove that two states are behaviourally equivalent by showing that they are bisimilar, since $x \sim y$ implies $x \sim_b y$. This proof principle is complete iff T preserves weak pullbacks.

An important property of bisimulations is that they can be characterised using relation lifting (see e.g. [128]). We explain this in some detail. Let $R \subseteq X_1 \times X_2$ be a relation with projections $\pi_1\colon R \to X_1$ and $\pi_2\colon R \to X_2$. By applying the functor, we obtain a pair of maps $\langle T(\pi_1), T(\pi_2) \rangle\colon T(R) \to T(X_1) \times T(X_2)$ whose image is a relation $Rel(T)(R) \subseteq T(X_1) \times T(X_2)$:

$$Rel(T)(R) \;=\; \{\langle T(\pi_1)(u), T(\pi_2)(u) \rangle \mid u \in T(R)\}. \tag{2.1}$$

It easily follows that a relation $R \subseteq X_1 \times X_2$ is a T-bisimulation between T-coalgebras $\langle X_1, \xi_1 \rangle$ and $\langle X_2, \xi_2 \rangle$ if and only if

$$\text{for all } \langle x_1, x_2 \rangle \in R : \langle \xi_1(x_1), \xi_2(x_2) \rangle \in Rel(T)(R). \tag{2.2}$$

For example, when considering Kripke frames ($\mathcal{P}$-coalgebras), (2.2) yields the well known back-and-forth bisimulation conditions, cf. Subsection 2.4.3. When T preserves weak pullbacks, $Rel(T)$ is the unique extension of T to a functor on the category Rel of sets and relations. We refer to [128] for more details.

2.4 Examples

As already mentioned in the introduction, many structures can be modelled as coalgebras (see e.g. [129, 69]): streams, trees, (labelled) transition structures, classical deterministic and nondeterministic automata, Mealy and Moore machines, Kripke frames and models, topological spaces [49] and various types of

probabilistic systems [16]. Further examples can be found in [136, 137]. The examples of streams, deterministic automata and Kripke frames are particularly relevant for the work in this thesis. We now summarise their coalgebraic modelling in order to provide the reader with a compact overview, and to create a point of reference for these facts.

2.4.1 Streams

Let A be a fixed set. A coalgebra for the functor $A \times (-)$ is a pair $\langle X, \langle o, d\rangle\rangle$ where $\langle o, d\rangle \colon X \to A \times X$ assigns to each state $x \in X$ an output value $o(x) \in A$ and a next state $d(x) \in X$. An $A \times (-)$-coalgebra can be visualised as a directed graph where edges are labelled with elements from A, and each state has exactly one outgoing edge.

The final $A \times (-)$-coalgebra consists of the set A^ω of streams over A together with the familiar head (hd) and tail (tl) functions: $\langle \mathrm{hd}, \mathrm{tl}\rangle \colon A^\omega \to A \times A^\omega$. The final map ϕ assigns to a state x the stream of outputs that is generated on the unique path starting from x (cf. [130]).

An $A \times (-)$-bisimulation between $\langle X_1, \langle o_1, d_1\rangle\rangle$ and $\langle X_2, \langle o_2, d_2\rangle\rangle$ is a relation $R \subseteq X_1 \times X_2$ such that for all $\langle x_1, x_2\rangle \in R$: $o_1(x_1) = o_2(x_2)$ and $\langle d_1(x_1), d_2(x_2)\rangle \in R$. An $A \times (-)$-bisimulation on $\langle A^\omega, \langle \mathrm{hd}, \mathrm{tl}\rangle\rangle$ is also called a stream bisimulation.

Suppose now that $A = \mathbb{N}$, the set of natural numbers. We can define the elementwise addition $+_\omega$ of two streams $\alpha, \beta \in \mathbb{N}^\omega$ by coinduction: We define an $\mathbb{N} \times (-)$-coalgebra structure ξ_+ on $\mathbb{N}^\omega \times \mathbb{N}^\omega$ such that for $\langle \alpha, \beta\rangle \in \mathbb{N}^\omega \times \mathbb{N}^\omega$ we obtain $\phi(\langle \alpha, \beta\rangle) = \alpha +_\omega \beta$. This is done by taking $\xi_+(\langle \alpha, \beta\rangle) = \langle \mathrm{hd}(\alpha) + \mathrm{hd}(\beta), \langle \mathrm{tl}(\alpha), \mathrm{tl}(\beta)\rangle\rangle$. It is now possible to prove by coinduction that $(\alpha_0, \alpha_1, \dots) +_\omega (\beta_0, \beta_1, \dots) = (\alpha_0 + \beta_0, \alpha_1 + \beta_1, \dots)$. Coinduction on streams has been extensively explored in [130, 131].

2.4.2 Deterministic automata

Let A be a set and $2 = \{0, 1\}$. A deterministic automaton with input alphabet A can be seen as a map $\langle o, d\rangle \colon Q \to 2 \times Q^A$, where Q is the set of states, $d \colon Q \to Q^A$ is the next state function, and the set of final states $F \subseteq Q$ is given by its characteristic function $o \colon Q \to 2$, i.e., $q \in F$ iff $o(q) = 1$. Deterministic automata are thus coalgebras for the functor $2 \times (-)^A$, which maps $f \colon Q_1 \to Q_2$ to the function $2 \times f^A \colon 2 \times Q_1^A \to 2 \times Q_2^A$ given by $(2 \times f^A)(\langle b, s\rangle) = \langle b, f \circ s\rangle$.

A function $f \colon Q_1 \to Q_2$ is a $2 \times (-)^A$-coalgebra morphism from $\langle Q_1, \langle o_1, d_1\rangle\rangle$ to $\langle Q_2, \langle o_2, d_2\rangle\rangle$ if for all $q \in Q_1$ and for all $a \in A$: $o_1(q) = o_2(f(q))$ and $f(d_1(q)(a)) = d_2(f(q))(a)$. This condition amounts to the known definition of a morphism (or state mapping) between deterministic automata, see e.g. [41]. A bisimulation between two $2 \times (-)^A$-coalgebras $\langle Q_1, \langle o_1, d_1\rangle\rangle$ and $\langle Q_2, \langle o_2, d_2\rangle\rangle$ is

a relation $R \subseteq Q_1 \times Q_2$ such that for all $\langle q_1, q_2 \rangle \in R$: $o_1(q_1) = o_2(q_2)$ and for all $a \in A$: $\langle d_1(q_1)(a), d_2(q_2)(a) \rangle \in R$.

The final $2 \times (-)^A$-coalgebra (cf. [126]) consists of the set $\mathcal{P}(A^*)$ of all languages over A together with the maps $E\colon \mathcal{P}(A^*) \to 2$ and $D\colon \mathcal{P}(A^*) \to \mathcal{P}(A^*)$, which are defined for all languages $L \subseteq A^*$ by $E(L) = 1$ iff $\varepsilon \in L$, and for all $a \in A$, we put $D(L)(a) = \{w \in A^* \mid aw \in L\}$. The set $D(L)(a)$ is also known as the language derivative of L with respect to a. The final map sends a state q to the set of words accepted from q. More precisely, given a $2 \times (-)^A$-coalgebra $\langle Q, \langle o, d \rangle \rangle$, the final map $L\colon Q \to \mathcal{P}(A^*)$ is defined by $L(q) = \{w \in A^* \mid d(q)(w) \in F\}$, where $d(q)(-)$ has been extended from letters to words in the canonical way.

2.4.3 Kripke frames

A Kripke frame is a pair $\langle X, R \rangle$ where X is a set and R is a binary relation on X. For $x \in X$, the set of R-successors of x is denoted by $R[x] = \{y \in X \mid \langle x, y \rangle \in R\}$. A Kripke frame $\langle X, R \rangle$ is a $\mathcal{P}$-coalgebra $\xi\colon X \to \mathcal{P}(X)$ by defining $\xi(x) = R[x]$. A $\mathcal{P}$-coalgebra morphism $f\colon \langle X_1, \xi_1 \rangle \to \langle X_2, \xi_2 \rangle$ is a function $f\colon X_1 \to X_2$ such that for all $x \in X_1$: $f[\xi_1(x)] = \xi_1(f(x))$, which is easily seen to be equivalent with the definition of a Kripke frame morphism, usually called a bounded morphism.

Using the relation lifting characterisation in (2.2), we find that a relation $Z \subseteq X_1 \times X_2$ with projections $\pi_i\colon Z \to X_i$, $i \in \{1, 2\}$, is a $\mathcal{P}$-bisimulation between Kripke frames $\langle X_1, R_1 \rangle$ and $\langle X_2, R_2 \rangle$, iff for all $\langle x_1, x_2 \rangle \in Z$: $\langle R_1[x_1], R_2[x_2] \rangle = \langle \pi_1[U], \pi_2[U] \rangle$ for some $u \in \mathcal{P}(Z)$. It can easily be verified that a $\mathcal{P}$-bisimulation satisfies the back-and-forth conditions for Kripke bisimulations. Assume now that Z is a Kripke bisimulation, and $\langle x_1, x_2 \rangle \in Z$. Taking $U := \pi_1^{-1}[R_1[x_1]] \cap \pi_2^{-1}[R_2[x_2]]$, we have: $y_1 \in \pi_1[U]$ iff $y_1 \in R_1[x_1]$ and there is a $y_2 \in R_2[x_2]$ such that $\langle y_1, y_2 \rangle \in Z$. Hence $\pi_1[U] \subseteq R_1[x_1]$ and $\pi_2[U] \subseteq R_2[x_2]$ are immediate, and the converse inclusions follow from the back-and-forth conditions on Z.

The final $\mathcal{P}$-coalgebra does not exist. This is a consequence of Lambek's lemma [90], which says that in a final coalgebra $\langle X, \xi \rangle$ the map ξ must be an isomorphism, together with the fact that there is no set X such that $X \cong \mathcal{P}(X)$. If we restrict $\mathcal{P}$ to its finitary part $\mathcal{P}_\omega$, which maps a set X to all its finite subsets, we obtain a final $\mathcal{P}_\omega$-coalgebra consisting of the set of finitely branching, possibly infinite, strongly extensional trees [160]. $\mathcal{P}_\omega$-coalgebras are the same as image-finite Kripke frames. The final $\mathcal{P}$-coalgebra exists in the category of classes [1], though. Here $\mathcal{P}$ is the functor which maps a class X to the collection of its subsets.

Chapter 3

Coalgebraic synthesis of Mealy machines

3.1 Introduction

Mealy machines were introduced in [96] to model the behaviour of sequential circuits. As a basic model of synchronous, ongoing behaviour, Mealy machines are now used much more generally in the modelling and specification of reactive systems [59, 135, 145, 158]. We are mainly motivated by their importance in digital circuit design [72, 76], and the use of Boolean and binary arithmetic in the specification of circuit behaviour. Digital hardware synthesis refers to the process of transforming a behavioural specification in the form of a Mealy machine into a sequential circuit with the specified behaviour. We use the term *Mealy synthesis* to refer to the process of transforming a syntactic specification s into a finite Mealy machine with the specified behaviour, also called a realisation of s. Combining Mealy synthesis with hardware synthesis thus yields a complete construction procedure from a formal algebraic specification to a digital circuit. In hardware synthesis, the number of registers of the resulting circuit depends on the number of states of the Mealy machine. Although various heuristics exist for producing a small circuit, it is generally of interest to start with a minimal Mealy machine.

With *coalgebraic synthesis* we refer to a construction method where the set of specifications is equipped with coalgebraic structure such that given a specification s, a coalgebra with the behaviour specified by s can be computed as a finite quotient of the subcoalgebra generated by s. The first steps towards coalgebraic synthesis of Mealy machines from 2-adic and mod-2 specifications were made by Rutten [132]. We now briefly summarise Rutten's results.

It is well-known that Mealy machines with input alphabet A and output alphabet B can be seen as coalgebras for the Set-functor $\mathcal{M}(X) = (B \times X)^A$. Rutten [132] shows that the set of so-called causal stream functions $f\colon A^\omega \to B^\omega$ is a final $\mathcal{M}$-coalgebra when equipped with the operations of initial output and stream function derivative. Bitstream operations in 2-adic or mod-2 arithmetic can be defined in a natural way using coinduction for bitstreams. Rutten

also shows in [132] that these operations give rise to causal bitstream functions $f\colon 2^\omega \to 2^\omega$ and to a Mealy coalgebra structure on the corresponding set of arithmetic expressions such that the final map yields the intended bitstream function semantics. Specifications in 2-adic and mod-2 arithmetic in general give rise to infinite-state behaviours, but Rutten defines a class of so-called rational specifications and shows that they can be realised by a finite Mealy machine. The only part that is missing in [132] in order to make coalgebraic synthesis work is a decidable congruence on the set of rational specifications which will ensure termination.

Our contributions to the coalgebraic Mealy synthesis method presented here are as follows. In [55] we showed that equivalence of rational specifications in 2-adic and mod-2 arithmetic can be effectively determined. The resulting coalgebraic synthesis algorithm therefore produces a minimal Mealy machine. We implemented this synthesis procedure for specifications in both 2-adic and mod-2 arithmetic. We gave an upper bound on the number of states in the minimal Mealy machine realising a rational 2-adic specification. The current chapter is an extension and improvement of the work in [55]. The main additions are: (i) Rational mod-2 bitstream functions and expressions are treated in detail. (ii) A slightly generalised definition of rational 2-adic and mod-2 function specifications; a proof that these can be realised by finite Mealy machines; and upper bounds on the size of their minimal realisations. (iii) Examples showing that rational 2-adic and mod-2 specifications are not expressively complete for Mealy machines. (iv) A more detailed exposition of the specification languages and their formal algebraic and coalgebraic semantics. (v) A more detailed account of the method used to determine equivalence of expressions via reduction to normal form. (vi) A detailed analysis of the time complexity of our construction of Mealy machines from rational function specifications. In fact, the time complexity given in [55] is not correct.

The chapter is structured as follows. Section 3.2 contains a review of Mealy machines, their coalgebraic modelling and causal stream function semantics. Section 3.3 first provides some basic definitions related to bitstreams and numbers, and subsequently describes the algebraic structure on bitstreams obtained from the 2-adic and mod-2 operations. This section also contains the abovementioned definitions and results on rational 2-adic and mod-2 functions. Section 3.4 describes our implementation of the Mealy synthesis algorithm. First, we introduce data types and define the functions which give a Mealy structure to expressions, and we explain how equivalence of expressions is determined. The last subsections contains a high-level description of the our algorithm, and an illustration of the construction with an example. Section 3.5 contains a detailed analysis of the time complexity of our algorithm. Finally, we conclude and discuss related work in Section 3.6.

3.2 Mealy machines

Assume two sets A and B are given. A Mealy machine with input in A and output in B is a deterministic state machine which in its current state, when supplied with an input letter from A, produces an output letter in B and moves to the next state. Formally, a *Mealy machine* is a 4-tuple (Q, o, d, q_0) where Q is a set of states, $o\colon Q \times A \to B$ is an output function, $d\colon Q \times A \to Q$ is a next-state function, and $q_0 \in Q$ is the initial state. A Mealy machine is *finite*, if A, B and Q are finite. Mealy machines are usually finite by definition, but we view finite Mealy machines as a special subclass, since the coalgebraic modelling (given in the next subsection) does not require such finiteness assumptions. A *binary Mealy machine* is a Mealy machine in which $A = B = 2 = \{0, 1\}$. Sometimes o and d are allowed to be partial functions (with the same domain), but we assume that o and d are well-defined for all states and inputs. It will be convenient to think about o and d in their curried forms $o\colon Q \to B^A$ and $d\colon Q \to Q^A$, and to ease notation we will write o_q and d_q rather than $o(q)$ and $d(q)$. When $o_q(a) = b$ and $d_q(a) = r$, we sometimes use the notation:

$$q \xrightarrow{a|b} r\,.$$

3.2.1 Mealy coalgebras

Using pairing and currying, we see that o and d correspond uniquely to a single map $\langle o, d\rangle\colon Q \times A \to B \times Q \;\cong\; t\colon Q \to (B \times Q)^A$, i.e., $t(q)(a) = \langle o_q(a), d_q(a)\rangle$. We call a map $t\colon Q \to (B \times Q)^A$ a *Mealy transition structure*. Clearly, a Mealy transition structure is a coalgebra for the polynomial functor $\mathcal{M}\colon \mathsf{Set} \to \mathsf{Set}$ which maps a set X to $\mathcal{M}(X) = (B \times X)^A$ and a function $g\colon X \to Y$ to $\mathcal{M}(g) = (id_B \times g)^A$. We refer to $\mathcal{M}$-coalgebras as *Mealy coalgebras*. Given a Mealy coalgebra $\mathbb{M} = (Q, t)$, we denote by $|\mathbb{M}|$ the cardinality $|Q|$ of the state set Q. Thus, a *Mealy machine* is simply a pointed Mealy coalgebra (Q, t, q_0).

Instantiating the general definitions of coalgebra morphism and bisimulation to Mealy coalgebras we obtain the following. Let $\mathbb{M} = (Q, t)$ and $\mathbb{M}' = (Q', t')$ be Mealy coalgebras. A function $g\colon Q \to Q'$ is a *Mealy morphism* from $\mathbb{M}$ to $\mathbb{M}'$, if for all $q \in Q$: $\mathcal{M}(g)(t(q)) = t'(g(q))$, which means that for all $a \in A$, if $t(q)(a) = \langle b, r\rangle$ then $t'(g(q))(a) = \langle b, g(r)\rangle$, i.e.:

$$q \xrightarrow{a|b} r \implies g(q) \xrightarrow{a|b} g(r)\,.$$

In terms of output and next state functions, g is a Mealy morphism if $o_q(a) = o'_{g(q)(a)}$ and $g(d_q(a)) = d'_{g(q)}(a)$, which is easily seen to coincide with the notion of a state mapping from [41]. Similarly, we find that a relation $R \subseteq Q \times Q'$ is a *Mealy bisimulation*, if for all $\langle q, q'\rangle \in R$ and all $a \in A$: $o_q(a) = o'_{q'}(a)$ and $\langle d_q(a), d'_{q'}(a)\rangle \in R$.

3.2.2 Causal stream functions

We will need the following basic definitions on streams. Given a set S, the set of streams over S is $S^\omega = \{\alpha \mid \alpha\colon \mathbb{N} \to S\}$, where $\mathbb{N}$ is the set of natural numbers. A stream is often denoted as an infinite sequence $\alpha = (\alpha(0), \alpha(1), \alpha(2), \dots)$, and we will also use ω-word notation. For example, $11(01)^\omega$ denotes the stream $(1, 1, 0, 1, 0, 1, \dots)$. For $\alpha \in S^\omega$, the *initial value* (head) of α is $\alpha(0)$, the *(stream) derivative* (tail) of α is $\alpha' = (\alpha(1), \alpha(2), \alpha(3), \dots)$ and for $s \in S$, $s : \alpha$ is the stream $(s, \alpha(0), \alpha(1), \dots)$. For $k \in \mathbb{N}$, $\alpha^{(k)}$ is the k'th derivative of α defined inductively by $\alpha^{(0)} = \alpha$ and $\alpha^{(k+1)} = (\alpha^{(k)})'$. We write $\#(\alpha) = |\{\alpha^{(k)} \mid k \in \mathbb{N}\}|$ for the number of distinct derivatives of α.

In order to describe the behaviour of Mealy machines we extend the output and next-state functions at a state q to maps $o_q\colon A^* \to B$ and $d_q\colon A^* \to Q$ in the following inductive manner:

$$\begin{array}{lcllcl} o_q(\varepsilon) & = & \varepsilon, & o_q(wa) & = & o_{d_q(w)}(a), \\ d_q(\varepsilon) & = & q, & d_q(wa) & = & d_{d_q(w)}(a). \end{array} \tag{3.1}$$

Suppose we are given a state q in a Mealy coalgebra $\mathbb{M} = (Q, t)$. We can then consider the transformation of input streams to output streams computed by $\mathbb{M}$ when starting in state q. For illustration, if the input stream is $(a_0, a_1, a_2, \dots) \in A^\omega$, then the output stream $(b_0, b_1, b_2, \dots) \in B^\omega$ is obtained as:

$$q \xrightarrow{a_0|b_0} q_1 \xrightarrow{a_1|b_1} \quad \cdots \quad \xrightarrow{a_k|b_k} q_{k+1} \quad \cdots$$

This stream function is called the *(Mealy) behaviour of* q, denoted $\mathrm{Beh}(q)\colon A^\omega \to B^\omega$, and it is inductively defined for $\alpha \in A^\omega$ and $k \geq 0$ by:

$$\mathrm{Beh}(q)(\alpha)(k) \;=\; o_q(\alpha(0) \dots \alpha(k)). \tag{3.2}$$

This definition makes it clear that for fixed q, $\mathrm{Beh}(q)(\alpha)(k)$ is determined by $\alpha(0), \dots, \alpha(k)$. Stream functions with this property are called causal. Formally, $f\colon A^\omega \to B^\omega$ is *causal* if for all $\alpha, \beta \in A^\omega$ and for all $n \in \mathbb{N}$,

$$\text{if for all } k \in \{0, \dots, n\} : \alpha(k) = \beta(k) \;\text{ then }\; f(\alpha)(n) = f(\beta)(n).$$

A causal stream function f is called *realisable* if there exists a finite Mealy machine $\mathbb{M} = (Q, t, q)$ such that $\mathrm{Beh}(q) = f$, in which case we say that $\mathbb{M}$ is a *realisation* of f.

We now demonstrate that the set of causal stream functions itself carries the structure of a Mealy coalgebra via the notions of initial output and (stream function) derivative.

3.2.1. Definition. Let $\Gamma = \{f\colon A^\omega \to B^\omega \mid f \text{ is causal}\}$, $f \in \Gamma$ and $a \in A$. The *initial output of f (on input a)* is defined as

$$f[a] := f(a{:}\alpha)(0) \qquad \text{for any } \alpha \in A^\omega.$$

The *derivative of f (on input a)* is the function $f{\cdot}a\colon A^\omega \to B^\omega$ defined by taking

$$(f{\cdot}a)(\alpha) := f(a{:}\alpha)' \qquad \text{for all } \alpha \in A^\omega.$$

We define a Mealy transition structure $\gamma : \Gamma \to (B \times \Gamma)^A$ by

$$\gamma(f)(a) = \langle f[a], f{\cdot}a\rangle \qquad \text{i.e.} \qquad f \xrightarrow{a|f[a]} f{\cdot}a\,. \qquad \lhd$$

Note that in Definition 3.2.1, $f[a]$ is well-defined, since f is causal: $f[a]$ depends only on $(a : \alpha)(0) = a$. Similarly, it is easy to show that the derivative of a causal stream function is again causal: $(f{\cdot}a)(\alpha)(k) = f(a : \alpha)'(k) = f(a{:}\alpha)(k+1)$, which depends only on $a, \alpha(0), \ldots, \alpha(k)$.

The initial output and derivative functions can be extended from letters to words over A as described in (3.1). The elements of the set $\{f{\cdot}w \mid w \in A^*\}$ are called the *derivatives of f*, and we denote with $\#(f)$ the number of derivatives of f. For reasons that will become clear from Theorem 3.2.2 below, we call f *finite-state*, if $\#(f) < \omega$, and *infinite-state* if $\#(f) = \omega$. In other words, f is finite-state iff f is realisable.

Rutten [132] showed that the operations of initial output and stream function derivative are universal in the sense that they make the set of causal stream functions into a final Mealy coalgebra.

3.2.2. Theorem ([132]). *The Mealy coalgebra (Γ, γ) (from Definition 3.2.1) is a final Mealy coalgebra. In particular, for every Mealy coalgebra (Q, t), the behaviour map* $\mathrm{Beh} : Q \to \Gamma$ *as defined in (3.2) is the unique Mealy morphism from (Q, t) to (Γ, γ).*

Proof. The proof is straightforward, but we include it for completeness' sake. Let (Γ, γ) be as stated, and let (Q, t) be an arbitrary Mealy coalgebra. We first verify that $\mathrm{Beh} : Q \to \Gamma$ is a Mealy homomorphism. So let $q \in Q$ and $\alpha \in A^\omega$ be arbitrary. We have:

$$\mathrm{Beh}(q)[a] = \mathrm{Beh}(q)(a{:}\alpha)(0) = o_q(a),$$

and by letting $q_0 = d_q(a)$ and $q_{i+1} = d_q(\alpha(i))$ for all $i \geq 0$, we have

$$\begin{aligned}
(\mathrm{Beh}(q){\cdot}a)(\alpha) &= (\mathrm{Beh}(q)(a{:}\alpha))' \\
&= (o_q(a), o_{d_q(a)}(\alpha(0)), o_{q_1}(\alpha(1)), \ldots)' \\
&= (o_{d_q(a)}(\alpha(0)), o_{q_1}(\alpha(1)), \ldots) \\
&= \mathrm{Beh}(d_q(a))(\alpha).
\end{aligned}$$

Hence $\mathrm{Beh}(q)\cdot a = \mathrm{Beh}(d_q(a))$. To see that $\mathrm{Beh} : Q \to \Gamma$ is unique, suppose $g : Q \to \Gamma$ is also a Mealy morphism. That is, for all $q \in Q$ and all $a \in A$,

$$\begin{array}{rcl} g(q)[a] & = & o_q(a) = \mathrm{Beh}(q)[a], \\ g(q)\cdot a & = & g(d_q(a)). \end{array} \tag{3.3}$$

We will show that the relation $R \subseteq B^\omega \times B^\omega$ defined by

$$R := \{\langle g(q)(\alpha), \mathrm{Beh}(q)(\alpha)\rangle \mid q \in Q, \alpha \in A^\omega\}.$$

is a stream bisimulation. It then follows by coinduction on streams in B^ω that for all $q \in Q$ and all $\alpha \in A^\omega$, $g(q)(\alpha) = \mathrm{Beh}(q)(\alpha)$, i.e., $g(q) = \mathrm{Beh}(q)$.

The initial values of $g(q)(\alpha)$ and $\mathrm{Beh}(q)(\alpha)$ agree, since

$$g(q)(\alpha)(0) = g(q)[\alpha(0)] \overset{(3.3)}{=} \mathrm{Beh}(q)[\alpha(0)] = \mathrm{Beh}(q)(\alpha)(0).$$

Also from the assumption that g and Beh are Mealy morphisms, we get,

$$\begin{array}{rcccl} g(q)(\alpha)' & = & (g(q)\cdot\alpha(0))(\alpha') & = & g(d_q(\alpha(0)))(\alpha'), \quad \text{and} \\ \mathrm{Beh}(q)(\alpha)' & = & (\mathrm{Beh}(q)\cdot\alpha(0))(\alpha') & = & \mathrm{Beh}(d_q(\alpha(0)))(\alpha') \end{array}$$

Hence the stream derivatives are again R-related, and we conclude that R is a stream bisimulation. QED

3.2.3. Remark. We will see in Chapter 4 that the final Mealy coalgebra can also be characterised by the set of length- and prefix-preserving functions of type $f\colon A^* \to B^*$. In this chapter, we work with the stream function semantics, since we will specify binary Mealy machines using the algebraic structures on bitstreams described in section 3.3. Yet another, perhaps even better known, representation of the final Mealy coalgebra is obtained by taking the set of functions $\{f\colon A^+ \to B\}$ as the carrier set. ◁

From the finality of (Γ, γ) it follows that for any causal $f \in \Gamma$, the subcoalgebra $\langle f\rangle$ generated by f in (Γ, γ) is a minimal Mealy coalgebra with behaviour f, and the states of $\langle f\rangle$ are the derivatives of f.

3.2.4. Remark. Causal stream function semantics were studied by G.N. Raney already in 1958: Raney introduces in [118] the notion of stream function derivative, and shows that derivatives of causal stream functions are again causal. Raney refers to derivatives of f as the *states of* f, but he does not explicitly show that the set of causal stream functions can be given a Mealy structure. Raney's [118] also contains results on compositions of causal stream functions, and their number of derivatives. We state some of these results below, since they are of interest to our work in sections 3.3.3 and 3.3.4 where we investigate the size of minimal realisations of rational bitstream functions. Other work by Raney [119, 120] on formal power series, generating functions and automata is closely related to the stream calculus described in Rutten [130]. ◁

3.2.5. PROPOSITION. *([118]). Let $f, g\colon A^\omega \to A^\omega$ be finite-state, causal stream functions.*

1. *The composition $f \circ g$ is causal, and $\#(f \circ g) \leq \#(f) \cdot \#(g)$.*
2. *If f and g are inverses, i.e., $f \circ g = g \circ f = id_{A^\omega}$, then: $\#(f) = \#(g)$.*

3.3 Bitstream algebras

In this section we describe the algebraic structures on the set of bitstreams which will be used as input specifications to our synthesis procedure. The first bitstream algebra to be presented is based on the arithmetic operations on 2-adic numbers [75, 47]. The motivation for studying this structure is its relevance for sequential binary arithmetic and digital circuits. Little literature seems to be available on this subject, with the exception of the work by Vuillemin (cf. [155, 156, 157]). The second bitstream algebra is based on addition modulo-2, and it is also motivated by its connection to digital circuits, and switching theory [76], in particular, to the theory and design of linear circuits.

3.3.1 Bitstreams and numbers

The integers are denoted by $\mathbb{Z}$, and the rational numbers by $\mathbb{Q}$. The absolute value of a rational number x is written $|x|$. We use the notation 2 to denote both the set $\{0, 1\}$ and the integer 2. The context should make clear which reading is intended. Elements of 2^ω are called bitstreams. A bitstream $\alpha \in 2^\omega$ is *eventually periodic*, if there exist $k, n \in \mathbb{N}$ such that $k < n$ and $\alpha^{(k)} = \alpha^{(n)}$. A bitstream $\alpha \in 2^\omega$ is *eventually constant* if there exists a $k \in \mathbb{N}$ such that $\alpha^{(k)} = \alpha^{(k+1)}$, that is, α ends with a tail of 0's or a tail of 1's, and hence $\#(\alpha) = 1 + \min\{k \in \mathbb{N} \mid \alpha^{(k)} = \alpha^{(k+1)}\}$.

We will define bitstreams and bitstream operations coinductively using the fact that 2^ω equipped with the operations initial value and stream derivative is a final coalgebra for the Set-functor $2 \times (-)$ (cf. Subsection 2.4.1). Following Rutten [130], coinductive definitions of bitstreams will be presented in the form of stream differential equations. In a system of *stream differential equations* streams are defined by specifying their initial value stream derivative. Under certain restrictions on the format, a system of stream differential equations is guaranteed to have a unique solution. A simple example is given by the bitstream $(1, 0, 1, 0, \dots)$ which can be defined as the unique solution in x of the system consisting of the stream differential equations:

$$\begin{array}{rclcrcl} x(0) & = & 1, & & x' & = & y, \\ y(0) & = & 0, & & y' & = & x. \end{array}$$

The set of rational numbers with odd denominator

$$\hat{\mathbb{Q}} = \{n/(2m+1) \mid n, m \in \mathbb{Z}\}$$

will play a significant role in the specification of 2-adic arithmetic. An element $q \in \hat{\mathbb{Q}}$ can be represented as a bitstream by taking its infinitary binary (2-adic) expansion $\mathrm{Bin}(q) \in 2^\omega$. If q is a natural number, then $\mathrm{Bin}(q)$ is just the binary representation of q followed by a tail of 0's. The bitstream $\mathrm{Bin}(-q)$ is an infinitary version of the two's complement of $\mathrm{Bin}(q)$. In general, it is known that for $q \in \hat{\mathbb{Q}}$, $\mathrm{Bin}(q)$ is an eventually periodic bitstream (cf. [47]).

We define $\mathrm{Bin}(q)$ coinductively, that is, by defining an appropriate $2 \times (-)$-coalgebra structure $\xi \colon \hat{\mathbb{Q}} \to 2 \times \hat{\mathbb{Q}}$ we obtain $\mathrm{Bin} \colon \hat{\mathbb{Q}} \to 2^\omega$ as the final map. For $q \in \hat{\mathbb{Q}}$, let

$$\xi(q) \;=\; \langle \mathit{odd}(q), (q - \mathit{odd}(q))/2 \rangle, \tag{3.4}$$

where $\mathit{odd}(n/(2m+1)) = n \bmod 2$. This definition is equivalent to defining $\mathrm{Bin}(q)$ as the solution to the set of stream diffential equations (one for each $q \in \hat{\mathbb{Q}}$):

$$\mathrm{Bin}(q)(0) = \mathit{odd}(q), \quad \mathrm{Bin}(q)' = \mathrm{Bin}((q - \mathit{odd}(q))/2). \tag{3.5}$$

We illustrate this definition with a few examples.

3.3.1. Example. The bitstream of observations generated by the element $9 \in \langle \hat{\mathbb{Q}}, \xi \rangle$ can be illustrated as follows:

$$9 \xrightarrow{1} \tfrac{9-1}{2} = 4 \xrightarrow{0} \tfrac{4}{2} = 2 \xrightarrow{0} \tfrac{2}{2} = 1 \xrightarrow{1} 0 \xrightarrow{0} 0$$

This shows that $\mathrm{Bin}(9) = (1001)0^\omega$ and $\#(\mathrm{Bin}(9)) = 5$. Now look at the stream behaviour of $-4 \in \langle \hat{\mathbb{Q}}, \xi \rangle$.

$$-4 \xrightarrow{0} \tfrac{-4}{2} = -2 \xrightarrow{0} \tfrac{-2}{2} = -1 \xrightarrow{1} \tfrac{-2}{2} = -1$$

Hence $\mathrm{Bin}(-4) = (00)1^\omega$ and $\#(\mathrm{Bin}(-4)) = 3$. Similarly, we find for $\frac{-1}{5} \in \langle \hat{\mathbb{Q}}, \xi \rangle$:

$$\tfrac{-1}{5} \xrightarrow{1} \tfrac{-6/5}{2} = \tfrac{-3}{5} \xrightarrow{1} \tfrac{-8/5}{2} = \tfrac{-4}{5} \xrightarrow{0} \tfrac{-4/5}{2} = \tfrac{-2}{5} \xrightarrow{0} \tfrac{-2/5}{2} = \tfrac{-1}{5}$$

Hence $\mathrm{Bin}(\frac{-1}{5}) = (1100)^\omega$ and $\#(\mathrm{Bin}(\frac{-1}{5})) = 4$. ◁

In the remainder of this section, we will switch freely between the numeric semantics and the bitstream semantics of elements from $\hat{\mathbb{Q}}$. In particular, we will simply write q instead of $\mathrm{Bin}(q)$, when convenient, and use the terms *integer, positive, negative* and *odd* about bitstreams with the obvious meaning. The numeric interpretation of bitstreams will be useful in the definition and analysis of rational 2-adic functions.

3.3.2 Bitstream algebra basics

We now describe the two algebraic structures which we use in specifying causal bitstream functions. The (semantic) domain of both algebras is the set of bitstreams 2^ω, and each structure provides interpretations of the binary function symbols $+, \times$ and $/$, the unary function symbol $-$, and the constants $[0], [1]$ and X. The constants are in both cases interpreted as the following bitstreams:

$$[0] = (0, 0, 0, 0, \ldots), \quad [1] = (1, 0, 0, 0, \ldots), \quad X = (0, 1, 0, 0, \ldots).$$

As usual, we write X^n for the n-fold product $X \times \ldots \times X$. The remaining operations are specific to each algebraic structure, and they will be defined using stream differential equations and the usual Boolean operations $\wedge, \vee, \neg$ and $\oplus$ (*exclusive-or*) on 2. For $a, b \in 2$: $a \wedge b = \min\{a, b\}$, $a \vee b = \max\{a, b\}$, $\neg a = 1 - a$, and $a \oplus b = (a \wedge \neg b) \vee (\neg a \wedge b)$.

It will turn out to be convenient to treat $/$ as a primitive constructor, rather than define α/β as shorthand for $\alpha \times (1/\beta)$ (as in [132]). The multiplicative inverse (if it exists) of a bitstream α can be defined as the fraction $[1]/\alpha$. Each bitstream algebra is an *integral domain*, that is, a commutative ring with unity and no zero divisors, which means that $\alpha \times \beta = [0]$ implies $\alpha = [0]$ or $\beta = [0]$. Hence $+, \times, -, /, [0]$ and $[1]$ will obey the familiar laws of arithmetic.

We will use a number of standard arithmetic conventions in our meta-notation. For example, we often write $x - y$ instead of $x + (-y)$, and $\frac{x}{y}$ instead of x/y. Moreover, brackets are used to disambiguate expressions but they are not part of the syntax, and in order to minimise the use of them we assume that the binding strength of the operations in descending order is $-, \times, /, +$. For example, $[1] - X + [1] \times X^2 = [1] + (-X) + ([1] \times X^2)$. Finally, from now on we will use σ only to denote a *bitstream variable*, i.e., a syntactic object, whereas α and β will be used as our meta-notation for bitstreams or expressions.

3.3.3 The 2-adic operations

The 2-adic bitstream algebra $\mathbb{A}_{2adic}$ is the structure on 2^ω obtained by viewing bitstreams as *2-adic integers* [47]. A 2-adic integer is a power series of the form $\Sigma_{i=0}^{\infty} a_i 2^i$, where $a_i \in \{0, 1\}$ for all $i \in \mathbb{N}$, and we represent it with the bitstream $\alpha = (a_0, a_1, a_2, \ldots)$. The 2-adic integers have interesting mathematical properties but for our purposes we only use the fact that they form an integral domain with the following operations. The addition of 2-adic integers is an infinitary version of binary addition, that is, carry bits may be propagated indefinitely. Similarly, minus is an infinitary version of two's complement. For example, $(1, 0, 0, \ldots) + (1, 1, 1, \ldots) = (0, 0, 0, \ldots)$, which shows that $-[1] = (1, 1, 1, \ldots)$. 2-adic multiplication is the Cauchy product, i.e., for bitstreams α and β, and $n \in \mathbb{N}$, $(\alpha \times \beta)(n) = \Sigma_{i=0}^{n} \alpha(i) \wedge \beta(n - i)$, where Σ denotes 2-adic summation.

Formally, we define the 2-adic operations on bitstreams by the stream differential equations in Figure 3.1. The fact that this system of equations has a unique solution follows from results in [130].

derivative			initial value			condition
$(\alpha+\beta)'$	$=$	$\alpha'+\beta'+[\alpha(0)\wedge\beta(0)]$	$(\alpha+\beta)(0)$	$=$	$\alpha(0)\oplus\beta(0)$	
$(-\alpha)'$	$=$	$-(\alpha'+[\alpha(0)])$	$(-\alpha)(0)$	$=$	$\alpha(0)$	
$(\alpha\times\beta)'$	$=$	$\alpha'\times\beta+[\alpha(0)]\times\beta'$	$(\alpha\times\beta)(0)$	$=$	$\alpha(0)\wedge\beta(0)$	
$(\alpha/\beta)'$	$=$	$(\alpha'-[\alpha(0)]\times\beta')/\beta$	$(\alpha/\beta)(0)$	$=$	$\alpha(0)$	$\beta(0)=1$

Figure 3.1: 2-adic operations

The stream differential equations for $+$ and $\times$ can easily be understood from the above description of sum and product of 2-adic integers. The defining equation for $-$ is derived from that of $+$ and the requirement that $\alpha+(-\alpha)=[0]$ by taking initial value and derivative on both sides. For the initial value we get that $\alpha(0)\oplus(-\alpha)(0)=0$, hence $(-\alpha)(0)=\alpha(0)$. By taking derivatives we get $\alpha'+(-\alpha)'+[\alpha(0)]=[0]$, and hence $(-\alpha)'=-([\alpha(0)]+\alpha')$. The stream differential equations for α/β can be derived similarly. From the shape of the stream differential equations in Figure 3.1, it is easy to see that any function defined using the 2-adic operations is causal. We mention some useful identities in $\mathbb{A}_{2adic}$:

$$\begin{aligned} \alpha+\alpha &= X\times\alpha = (0,\alpha(0),\alpha(1),\ldots) && \text{for all } \alpha\in 2^\omega,\\ X^n &= (\underbrace{0,\ldots,0}_{n \text{ times}},1,0,0,\ldots) && \text{for all } n\in\mathbb{N}. \end{aligned}$$

For example, to prove the identity $\alpha+\alpha=X\times\alpha$ one can show that for any $\alpha\in 2^\omega$, the relation $R=\{\langle\alpha+\alpha, X\times\alpha\rangle\}\cup\{\langle\beta'+\beta'+[\beta(0)],\beta\rangle \mid \beta\in 2^\omega\}$ is a stream bisimulation, hence by stream coinduction $\alpha+\alpha=X\times\alpha$. These identities, especially $\alpha+\alpha=X\times\alpha$, are easy to remember when thinking of the stream X as the integer 2. In fact, the following lemma tells us that we can treat the 2-adic operations on bitstreams of the form $\mathrm{Bin}(q)$, $q\in\hat{\mathbb{Q}}$, as the well-known arithmetic operations on rational numbers.

3.3.2. Lemma. *The map* $\mathrm{Bin}\colon\hat{\mathbb{Q}}\to 2^\omega$ *defined in* (3.5) *is a homomorphism of integral domains* $\mathrm{Bin}\colon\hat{\mathbb{Q}}\to\mathbb{A}_{2adic}$.

Proof. One can easily show (using stream coinduction) that $\mathrm{Bin}(0)=[0]$, $\mathrm{Bin}(1)=[1]$, $\mathrm{Bin}(2)=X$ and Bin commutes with sum, minus, product and division. The details are left to the reader. QED

In the remainder of this section, we rely on Lemma 3.3.2 to prove that the rational 2-adic functions (defined below) are realisable, and to give an upper bound on the size of their minimal realisations.

3.3.3. Definition. Call $\alpha \in 2^\omega$ a *polynomial 2-adic bitstream* if $\alpha = c_0 + c_1 \times X + c_2 \times X^2 + \ldots + c_k \times X^k$, where $c_i \in \{-[1], [0], [1]\}$ for $0 \leq i \leq k$. A bitstream $\rho \in 2^\omega$ is a *rational 2-adic bitstream* if $\rho = \alpha/\beta$, where α and β are polynomial 2-adic bitstreams and $\beta(0) = 1$. A bitstream function $f\colon 2^\omega \to 2^\omega$ is a *rational 2-adic function*, if f is of the form $f(\sigma) = \rho_1 + \rho_2 \times \sigma$ where ρ_1, ρ_2 are rational 2-adic bitstreams and σ is a bitstream variable. ◁

3.3.4. Remark. In [132], Rutten defines rational 2-adic functions as functions of the form $f(\sigma) = \rho \times \sigma$, ρ rational, and shows that they have only finitely many derivatives. Rutten mentions that this also holds for the rational 2-adic functions of Definition 3.3.3 (and we prove this below). We use the word 'rational' since our rational 2-adic functions are not substantially more general than those of Rutten's, and they share the property of having only finitely many derivatives. This is yet another reason for their name, since this property parallels the well-known result that rational languages have only finitely many language derivatives (also called left quotients).

When proving in [132, Theorem 5.1] that functions of the form $f(\sigma) = \rho \times \sigma$, ρ rational, have only finitely many derivatives, Rutten uses the notion of degree of polynomial bitstreams which he defines as follows. The degree of a polynomial 2-adic bitstream $\alpha = c_0 + c_1 \times X + c_2 \times X^2 + \ldots + c_k \times X^k$ is $\deg(\alpha) = k$ if $c_k \neq [0]$. However, $\deg(\alpha)$ depends on the chosen representation of α. For example, the bistream $(1, 1, 1, \ldots)$ can be written as $-[1]$, which has degree 0, or as $([1] + X + \ldots X^k) - X^{k+1}$, which has degree $k+1$, for any $k \in \mathbb{N}$. So Rutten's argument does not directly lead to an unambiguous upper bound on $\#(f)$ for a rational 2-adic f. Instead of the degree of polynomials, our reasoning uses the number of derivatives which is a semantic property and hence independent of any algebraic representation. Moreover, we will use the numeric interpretation of the 2-adic operations, and it turns out to be convenient to work with a representation that only contains one occurrence of the division operation. ◁

3.3.5. Lemma. *A bitstream function $f\colon 2^\omega \to 2^\omega$ is a rational 2-adic (bitstream) function iff f is of the form:*

$$f(\sigma) = \frac{d + m \times \sigma}{n} \tag{3.6}$$

where d, m and n are integer bitstreams, n is odd, and σ is a bitstream variable.

Proof. Follows from $\mathrm{Bin}\colon \hat{\mathbb{Q}} \to \mathbb{A}_{2adic}$ being a homomorphism of integral domains, the 1-1 correspondence between integers and polynomial 2-adic bitstreams, and the identities of integral domains. QED

From now on we will think of rational 2-adic functions in the format of (3.6). We continue by giving a numeric characterisation of the derivatives of rational 2-adic functions.

3.3.6. Lemma (Numeric 2-adic derivatives). *Let $f\colon 2^\omega \to 2^\omega$ be a rational 2-adic stream function of the form:*

$$f(\sigma) = \frac{d + m \times \sigma}{n}$$

for integers d, m and n with n odd. For $a \in 2$, the stream function derivative $f{\cdot}a$ is given by:

$$(f{\cdot}a)(\sigma) = \frac{d_a + m \times \sigma}{n} \tag{3.7}$$

where (in the numeric interpretation)

$$d_0 = \begin{cases} \frac{1}{2}\, d & \text{if } d \text{ even} \\ \frac{1}{2}(d-n) & \text{if } d \text{ odd} \end{cases} \qquad d_1 = \begin{cases} \frac{1}{2}(d+m) & \text{if } d(0) = m(0) \\ \frac{1}{2}(d+m-n) & \text{if } d(0) \neq m(0) \end{cases}$$

Proof. Assume $f\colon 2^\omega \to 2^\omega$ is of the above form, and let $a \in 2$. Applying the definitions in Figure 3.1 on page 30 to determine $f{\cdot}a$ we get

$$\begin{aligned}
& (f{\cdot}a)(\sigma) \\
= \;& \left(\frac{d + m \times (a:\sigma)}{n}\right)' \\
= \;& \frac{(d + m \times (a:\sigma))' - [d(0) \oplus (m(0) \wedge a)] \times n'}{n} \\
\overset{(\dagger)}{=} \;& \frac{d' + ([a] \times m' + m \times \sigma) + [d(0) \wedge m(0) \wedge a] - [d(0) \oplus (m(0) \wedge a)] \times n'}{n} \\
= \;& \begin{cases} \dfrac{d' - [d(0)] \times n' + m \times \sigma}{n} & \text{if } a = 0 \\[2ex] \dfrac{d' + m' + [d(0) \wedge m(0)] - [d(0) \oplus m(0)] \times n' + m \times \sigma}{n} & \text{if } a = 1 \end{cases}
\end{aligned}$$

The rewrite step marked with (†) uses commutativity of $\times$. Hence we obtain the following equations for the d_a-value in (3.7):

$$\begin{aligned} d_0 &= d' - [d(0)] \times n' \text{ and} \\ d_1 &= d' + m' + [d(0) \wedge m(0)] - [d(0) \oplus m(0)] \times n'. \end{aligned}$$

The rest of the proof is now straightforward using (3.4) (p. 28). If d is even then $d_0 = d' = \frac{1}{2}d$, and if d is odd, we get:

$$d_0 = d' - n' = \frac{1}{2}(d-1) - \frac{1}{2}(n-1) = \frac{1}{2}(d-n).$$

When d and m are both odd, i.e. $d(0) = m(0) = 1$, we have

$$d_1 = d' + m' + 1 = \frac{1}{2}(d-1) + \frac{1}{2}(m-1) + 1 = \frac{1}{2}(d+m).$$

Finally, if d is odd, and m is even, then

$$d_1 = d' + m' - n' = \frac{1}{2}(d-1) + \frac{1}{2}m - \frac{1}{2}(n-1) = \frac{1}{2}(d+m-n).$$

QED

3.3.7. Example. We compute some of the derivatives of the rational 2-adic function $f(\sigma) = \frac{-2+7\cdot\sigma}{3}$:

$$\begin{aligned}
f\cdot 0 &= \tfrac{-2/2+7\cdot\sigma}{3} &&= \tfrac{-1+7\cdot\sigma}{3}\\
f\cdot 1 &= \tfrac{((-2+7)-3)/2+7\cdot\sigma}{3} &&= \tfrac{1+7\cdot\sigma}{3}\\
f\cdot 00 &= \tfrac{(-1-3)/2+7\cdot\sigma}{3} &&= \tfrac{-2+7\cdot\sigma}{3} = f\\
f\cdot 01 &= \tfrac{(-1+7)/2+7\cdot\sigma}{3} &&= \tfrac{3+7\cdot\sigma}{3}\\
f\cdot 10 &= \tfrac{(1-3)/2+7\cdot\sigma}{3} &&= \tfrac{-1+7\cdot\sigma}{3}\\
f\cdot 11 &= \tfrac{(1+7)/2+7\cdot\sigma}{3} &&= \tfrac{4+7\cdot\sigma}{3}
\end{aligned}$$

Continuing in this way, we find that the set of derivatives of f is:

$$\left\{\tfrac{d+7\times\sigma}{3} \mid d \in \{-2,-1,0,\ldots,6\}\right\}.$$

Further examples of derivatives of rational 2-adic functions are listed here:

$f(\sigma)$	Derivatives
$\frac{12+7\times\sigma}{3}$	$\left\{\frac{d+7\times\sigma}{3} \mid d \in \{-2,-1,0,\ldots,6,8,12\}\right\}$
$\frac{9+(-1)\times\sigma}{5}$	$\left\{\frac{d+-1\times\sigma}{5} \mid d \in \{-5,-4,-3,\ldots,2,4,9\}\right\}$
$\frac{4+8\times\sigma}{11}$	$\left\{\frac{d+8\times\sigma}{11} \mid d \in \{-10,-9,-8,\ldots,7\}\right\}$
$\frac{-2+(-3)\times\sigma}{3}$	$\left\{\frac{d+-3\times\sigma}{3} \mid d \in \{-5,-4,\ldots,-1\}\right\}$

◁

The above example and experimental results suggested that the derivatives of $f(\sigma) = \frac{d+m\times\sigma}{n}$ can be described in terms of d, n, m. We will now show that this is indeed the case.

3.3.8. Lemma. *Let $f(\sigma) = \frac{d+m\times\sigma}{n}$ be a rational 2-adic function with $m \neq 0$ and $n > 0$ odd. For all $w \in 2^*$, the stream function derivative $f{\cdot}w$ is of the form*

$$(f{\cdot}w)(\sigma) = \frac{d_w + m \times \sigma}{n} \tag{3.8}$$

where d_w is an integer such that

$$\min\{d, -n+1, -n+m+1\} \;\leq\; d_w \;\leq\; \max\{d, m-1, 0\}.$$

Proof. It is a consequence of Lemma 3.3.6 that the derivatives of f have the given format (3.8), since f is itself of the form required in Lemma 3.3.6, and hence so are all derivatives of f. We prove by induction on the length of $w \in 2^*$ that the numeric value d_w is in the given range. The base case ($w = \varepsilon$) is clear. To prove the inductive step we use the numeric interpretation of derivatives of rational 2-adic functions given in Lemma 3.3.6. To ease notation, let $l = \min\{d, -n+1, -n+m+1\}$ and $u = \max\{d, m-1, 0\}$. Note that $l \leq 0 \leq u$. Assume as induction hypothesis (IH) that $l \leq d_w \leq u$. Inequalities obtained from the induction hypothesis will be denoted by $\leq_{IH}$.

Induction step for d_{w0}: We first consider the case where d_w is even, and thus $d_{w0} = \frac{1}{2}d_w$. We have the following cases:

$$\begin{array}{ll} \text{if } d_w \geq 0\text{:} & l \;\leq\; 0 \;\leq\; \frac{1}{2}d_w \;\leq\; d_w \leq_{IH} u. \\ \text{if } d_w < 0\text{:} & l \;\leq_{IH}\; d_w \;<\; \frac{1}{2}d_w \;<\; 0 \;\leq\; u. \end{array}$$

Now if d_w is odd, then $d_{w0} = \frac{1}{2}(d_w - n)$. To prove the lower bound, we have

$$l \;\leq\; -n+1 \;\Rightarrow\; 2l \;\leq\; l-n+1 \;\leq_{IH}\; d_w - n + 1,$$

and since $d_w - n + 1$ is odd, it follows that $2l \leq d_w - n$ and hence $l \leq \frac{1}{2}(d_w - n)$. The upper bound follows easily from $d_w - n \;<\; d_w \;\leq_{IH}\; u$ which implies $\frac{1}{2}(d_w - n) \;\leq\; u$, since $u \geq 0$.

Induction step for d_{w1}: If $d_w(0) = m(0)$, then $d_{w1} = \frac{1}{2}(d_w + m)$. We first prove the lower bound. We have,

$$l \;\leq\; -n+m+1 \;\leq\; m \quad\Rightarrow\quad 2l \;\leq_{IH}\; d_w + m,$$

and hence $l \leq \frac{1}{2}(d_w + m)$. For the upper bound, we have

$$m-1 \;\leq\; u \quad\Rightarrow\quad d_w + m - 1 \;\leq_{IH}\; 2u,$$

and since $d_w(0) = m(0)$, $d_w + m - 1$ must be odd. It follows that $d_w + m \leq 2u$, which in turn implies $\frac{1}{2}(d_w + m) \leq u$.

If $d_w(0) \neq m(0)$, then $d_{w1} = d_w + m - n$. We know that $l \leq -n + m + 1$ and hence $2l \leq l - n + m + 1 \leq_{IH} d_w + m - n + 1$. Since $d_w + m - n + 1$ is odd, it follows that $2l \leq d_w + m - n$, and hence $l \leq \frac{1}{2}(d_w + m - n)$.

The upper bound is proven in a similar fashion. We have $u \geq m - 1$ which implies $2u \geq u + m - 1 \geq u + m - n \geq_{IH} d_w + m - n$, and it follows that $u \geq \frac{1}{2}(d_w + m - n)$. QED

The following theorem is now immediate.

3.3.9. THEOREM. *All rational 2-adic bitstream functions are realisable.*

PROOF. Let $f(\sigma) = \frac{d+m\times\sigma}{n}$ be a rational 2-adic function. If $m = 0$, then f is constant equal to the rational bitstream $\mathrm{Bin}(\frac{d}{n})$, which is eventually periodic and hence realisable. If $m \neq 0$ and $n > 0$, then the statement follows from Lemma 3.3.8. If $m \neq 0$ and $n < 0$, then we use the fact that the function $g(\sigma) = \frac{-d+-m\times\sigma}{-n}$ is realisable by the previous case, and observe that g is bisimilar with f (cf. Lemma 3.3.2). Hence f is also realisable. QED

It can readily be checked that the values of d_w observed for the functions from Example 3.3.7 vary exactly between the lower and upper bounds given by Lemma 3.3.8. Since the d_w-values are limited in range, we can derive an upper bound on the number of derivatives of a rational 2-adic function. In order to obtain an optimal bound for $f(\sigma) = \frac{d+m\times\sigma}{n}$, we may assume $n > 0$ and that d, m and n are coprime, i.e., the greatest common divisor of d, m and n, $\gcd(d, m, n)$, is 1. For suppose, d, m and n do not satisfy this requirement, then we can always divide with $\gcd(d, m, n)$ and multiply by -1 and the resulting function $\tilde{f}$ is bisimilar with f in the final Mealy coalgebra, hence by finality $f = \tilde{f}$.

3.3.10. THEOREM. *Let $f(\sigma) = \frac{d+m\times\sigma}{n}$ be a rational 2-adic function where d, m and n are integers such that $m \neq 0$, $n > 0$ is odd and $\gcd(d, m, n) = 1$. The number $\#(f)$ of derivatives of f (which equals the number of states in the minimal Mealy coalgebra $\langle f \rangle$ realising f) is bounded by the values listed in the following table:*

item	*case m*	*case d*	$\#(f) \leq$
1	$m > 0$	$d < -n + 1$	$\lvert d\rvert + \lvert m\rvert$
2	$m > 0$	$-n + 1 \leq d \leq m - 1$	$\lvert m\rvert + \lvert n\rvert - 1$
3	$m > 0$	$d > m - 1$	$\lvert d\rvert + \lvert n\rvert$
4	$m < 0$	$d < -n + m + 1$	$\lvert d\rvert$
5	$m < 0$	$-n + m + 1 \leq d < 0$	$\lvert m\rvert + \lvert n\rvert - 1$
6	$m < 0$	$d \geq 0$	$\lvert d\rvert + \lvert m\rvert + \lvert n\rvert$

PROOF. From Lemma 3.3.8 we know that there is a 1-1 correspondence between derivatives $f{\cdot}w$ and the d_w value occurring in the numerator of $f{\cdot}w$, and that the value of d_w is bounded, i.e., $l \leq d_w \leq u$, for the values of l and u listed in the table below. Hence $\#(f) = |\{d_w \mid w \in 2^*\}| \leq |\{l, l+1, \ldots, u\}| = u - l + 1$. The u-value in items 4 and 5 does not follow directly from Lemma 3.3.8, but using the numeric interpretation of Lemma 3.3.6 it is easy to see that, if $d < 0$ then also $d_w < 0$, for all $w \in 2^*$.

item	case m	case d	l	u
1	$m > 0$	$d < -n+1$	d	$m-1$
2	$m > 0$	$-n+1 \leq d \leq m-1$	$-n+1$	$m-1$
3	$m > 0$	$d > m-1$	$-n+1$	d
4	$m < 0$	$d < -n+m+1$	d	-1
5	$m < 0$	$-n+m+1 \leq d < 0$	$-n+m+1$	-1
6	$m < 0$	$d \geq 0$	$-n+m+1$	d

QED

3.3.11. REMARK. Experimental results suggest that the upper bound on $\#(f)$ given in Theorem 3.3.10 is also a lower bound when $\gcd(m, n) = 1$ and d is in the following range: (i) $-n \leq d \leq m$ if $m > 0$, and (ii) $-n+m+1 \leq d \leq 0$ if $m < 0$. This conjecture has been verified for $-50 \leq m \leq 50$ and $1 \leq n \leq 23$. However, at present we have no formal proof of this. Although, the production of d_w values can be described by a reasonably simple recurrence over the natural numbers (cf. Lemma 3.3.6), analysing the behaviour of such a recurrence is far from trivial[1]. ◁

3.3.12. REMARK. Using case-by-case arguments similar to those used in the proof of Lemma 3.3.8, it is possible to give an upper bound on the number of derivatives of $f(\sigma) = \frac{d+m\times\sigma}{n}$ in terms of the number of derivatives of d, m, n. The upper bound obtained in this way is $\#(f) \leq 2^{k+1}$ where $k = \max\{\#(d), \#(m), \#(n)\}$. This bound exceeds the upper bound from Theorem 3.3.10 by 2^{k-1}. To see this, observe that for functions of the form required for Theorem 3.3.10, it follows that $k \geq 2$ since $m \neq 0$ and $n > 0$. For $k \geq 2$, the values of d, m and n which maximise the upper bound from Theorem 3.3.10 are $d = n = 2^{k-1} - 1$ and $m = -2^{k-1}$. For these values we find that the difference between the two upper bounds is:

$$2^{k+1} - ((2^k - 2) + 2^{k-1}) = 2^{k+1} - (2^k + 2^{k-1}) + 2 = 2^{k-1} + 2. \qquad \triangleleft$$

[1] A famous example of a simple recurrence with complicated behaviour is given by the Collatz function $f(n) = n/2$ if n even, otherwise $f(n) = 3n + 1$. The Collatz conjecture says that for all natural numbers $n \geq 1$, the sequence $f(n), f(f(n)), f^3(n), \ldots$ will eventually reach 1. The conjecture has yet to be formally proved, although it has been verified by computers for start values up to $10 \cdot 2^{58}$.

3.3.13. Remark. Theorem 3.3.10 is an improvement on Raney's result in Proposition 3.2.5(1). A rational 2-adic function $f(\sigma) = \frac{d+m\times\sigma}{n}$ can be seen as a composition $f(\sigma) = (\mathrm{Div}_n \circ \mathrm{Add}_d \circ \mathrm{Mul}_m)(\sigma)$, where for $z \in \mathbb{Z}$:

$$\begin{aligned} \mathrm{Mul}_z(\sigma) &= z \times \sigma, \\ \mathrm{Add}_z(\sigma) &= z + \sigma, \\ \mathrm{Div}_z(\sigma) &= \sigma/z \quad (\text{if } z \text{ is odd}). \end{aligned}$$

A bound on the number of derivatives of these three basic 2-adic functions is obtained by instantiating Lemma 3.3.8 with the appropriate values for d, m and n. That is, for Mul_m, take $d = 0, n = 1$, and for Add_d, take $m = 1, n = 1$. From Prop. 3.2.5, we know that $\#(\mathrm{Div}_z) = \#(\mathrm{Mul}_z)$ for all odd z. We find:

$$\begin{aligned} \#(\mathrm{Mul}_z) &\leq \begin{cases} z & \text{if } z > 0 \\ |z| + 1 & \text{if } z \leq 0 \end{cases} \\ \#(\mathrm{Add}_z) &\leq |z| + 1, \\ \#(\mathrm{Div}_z) &= \#(Mul_z) \quad (\text{if } z \text{ is odd}). \end{aligned}$$

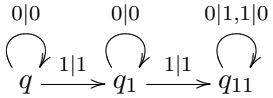
Experiments indicate that for Mul_z and Div_z this bound is tight, but also that the bound for $\#(\mathrm{Add}_z)$ can be improved to $2 \cdot (\#(z) - 1)$ if $(\#(z) \geq 2$ and $z > 0)$ or $\#(z) \geq 3$; and otherwise $\#(z) + 1$. Using the bound $2 \cdot (\#(z) - 1)$ for Add_z, it follows from Prop. 3.2.5 that if $m, n > 0$ and $\#(d) \geq 3$:

$$\#(f) \leq \#(\mathrm{Mul}_m) \cdot \#(\mathrm{Add}_d) \cdot \#(\mathrm{Div}_n) \leq 2 \cdot m \cdot n \cdot (\#(d) - 1).$$

For increasing values of m, n and d, this upper bound will clearly grow much faster than the upper bound from Theorem 3.3.10. ◁

Knowing that rational 2-adic functions are realisable (Theorem 3.3.9), the question naturally arises whether the converse holds. More precisely, is the behaviour of any finite Mealy machine a rational 2-adic function? This is answered in the negative by the following example.

3.3.14. Example. Consider the Mealy machine $\mathbb{M}$ depicted in the following diagram:



We will show that the stream function realised by the state q is *not* a 2-adic function of the form $f(\sigma) = \frac{d+m\times\sigma}{n}$ for integers d, m and n, with n odd. To this end, first observe that the stream functions realised by the states q_1 and q_{11}

are the rational 2-adic functions $g(\sigma) = -\sigma$ and $h(\sigma) = -1 - \sigma$, respectively. We prove this by showing that the relation $R = \{\langle q_1, g\rangle, \langle q_{11}, h\rangle\}$ is a Mealy bisimulation between $\mathbb{M}$ and the final Mealy coalgebra. We have for all $a \in 2$, $o(q_1)(a) = a = (-(a : \sigma))(0) = g[a]$. Using Lemma 3.3.6 or the definitions from Figure 3.1, it is easy to verify that $g{\cdot}0 = g$ and $g{\cdot}1 = h$, hence $\langle d(q_1)(a), g{\cdot}a\rangle \in R$ for all $a \in 2$. The bisimulation condition for $\langle q_{11}, h\rangle$ can be checked just as easily.

From Lemma 3.3.6, it follows that if $\mathrm{Beh}(q)$ is a rational 2-adic function, then $\mathrm{Beh}(q)$ can be written in the form $\mathrm{Beh}(q)(\sigma) = \frac{d+(-1)\times\sigma}{1}$, i.e., $m = -1$ and $n = 1$, as is the case for g and h. In order to arrive at a contradiction, suppose indeed that $f(\sigma) = \mathrm{Beh}(q)(\sigma) = \frac{d+(-1)\times\sigma}{1}$. Since $f{\cdot}0 = f$ and $f{\cdot}1 = g$, we see that $d_0 = d$ and $d_1 = 0$. The value d must be either even or odd. If d is even, then we have from Lemma 3.3.6:

$$\begin{aligned} d = d_0 = \tfrac{1}{2}d \quad &\Rightarrow \quad d = 0, \text{ and} \\ 0 = d_1 = \tfrac{1}{2}(d - 1 - 1) \quad &\Rightarrow \quad d = 2. \end{aligned}$$

So d cannot be even. But similarly, for d odd, we find that:

$$\begin{aligned} d = d_0 = \tfrac{1}{2}(d - 1) \quad &\Rightarrow \quad d = -1, \text{ and} \\ 0 = d_1 = \tfrac{1}{2}(d - 1) \quad &\Rightarrow \quad d = 1. \end{aligned}$$

Hence, there is no d such that $\mathrm{Beh}(q)(\sigma) = \frac{d+(-1)\times\sigma}{1}$. ◁

We conclude this subsection by showing that although any bitstream function specified using the 2-adic operations is causal, it is not necessarily realisable.

3.3.15. Example. *1. It is well-known that the 2-adic function $f(\sigma) = \sigma \times \sigma$ has infinite-state behaviour. To verify this, it is easy to show by induction on $n \in \mathbb{N}$ that the derivative of f with respect to 0^n is $(f{\cdot}0^n)(\sigma) = X^n \times (\sigma \times \sigma)$. For different n, these derivatives are clearly not equivalent, and hence $\langle f\rangle$ has infinitely many states.*

2. Another example known not to be finite-state is taking inverses (of rational streams with initial value 1). One can see this by considering the 2-adic function $g(\sigma) = [1]/([1] + X \times \sigma)$. Here one can show by induction that for all $n \geq 1$, $(g{\cdot}0^n)(\sigma) = (-X \times \sigma)/([1] + X^{n+1} \times \sigma)$, and again these derivatives are different for different n. ◁

The above example suggest that allowing powers σ^n for $n \notin \{0, 1\}$ in the specification will lead to infinite-state behaviour, and hence that rational 2-adic functions provide the most general form of realisable specifications in 2-adic arithmetic.

3.3.4 The mod-2 operations

The mod-2 bitstream algebra $\mathbb{A}_{mod2}$ is the integral domain structure on 2^ω obtained by taking element-wise addition modulo-2 as the sum operation, which we also denote by $\oplus$. Note that $\oplus\colon 2 \times 2 \to 2$ is nilpotent, i.e., for any $a \in 2$, $a \oplus a = 0$ and hence also for any $\alpha \in 2^\omega$, $\alpha \oplus \alpha = [0]$. The minus operation $\ominus$ is again defined as an additive inverse. From the nilpotency of $\oplus$ it follows that minus is identity: for all $\alpha \in 2^\omega : \ominus\alpha = \alpha$. Multiplication $\otimes$ is the Cauchy product with respect to $\oplus$-summation, and division $\oslash$ is defined such that $[1]\oslash\alpha$ is a multiplicative inverse of α under the condition $\alpha(0) = 1$. Fractions will usually be written in the form $\frac{\alpha}{\beta}$. The context should make clear when this notation should be read as a mod-2 fraction or a 2-adic fraction. The mod-2 operations on 2^ω are defined by the stream differential equations in Figure 3.2. As with the 2-adic operations, the syntactic shape of the equations gurantees that a unique solution exists [130].

derivative			initial value			condition
$(\alpha \oplus \beta)'$	$=$	$\alpha' \oplus \beta'$	$(\alpha \oplus \beta)(0)$	$=$	$\alpha(0) \oplus \beta(0)$	
$(\ominus\alpha)'$	$=$	$\ominus(\alpha')$	$(\ominus\alpha)(0)$	$=$	$\alpha(0)$	
$(\alpha \otimes \beta)'$	$=$	$(\alpha' \otimes \beta) \oplus [\alpha(0)] \otimes \beta'$	$(\alpha \otimes \beta)(0)$	$=$	$\alpha(0) \wedge \beta(0)$	
$(\alpha \oslash \beta)'$	$=$	$(\alpha' \oplus [\alpha(0)] \otimes \beta') \oslash \beta$	$(\alpha \oslash \beta)(0)$	$=$	$\alpha(0)$	$\beta(0) = 1$

Figure 3.2: mod-2 operations

The fact that $\mathbb{A}_{mod2}$ is an integral domain can be proved using stream coinduction. The straightforward proof is omitted here. Another way of seeing this uses the theory of formal power series, as follows. The mod-2 bitstream operations correspond to the operations obtained by viewing bitstreams as formal power series over the mod-2 ring $\mathbb{A}_2 = (2, \oplus, \wedge, id, 0, 1)$. In $\mathbb{A}_2$, sum is addition modulo 2, minus is identity, and multiplication is the Boolean operation $\wedge$. A bitstream $\alpha = (a_0, a_1, a_2 \ldots)$ is thus interpreted as the coefficients of the formal power series $a_0 + a_1 x + a_2 x^2 + \ldots$. From the theory of formal power series (see e.g. [159]), it follows that $\mathbb{A}_{mod2}$ is an integral domain. See also [130] for more information on the interesting connections between formal power series and streams.

The formal powers series perspective makes it easy to see that certain identities hold in $\mathbb{A}_{mod2}$. For example, the constant stream $X = (0, 1, 0, 0, \ldots)$ now plays the role of the formal variable x, and it can be verified (using bitstream coinduction) that, as in $\mathbb{A}_{2adic}$:

$$\begin{aligned} X \otimes \alpha &= (0, \alpha(0), \alpha(1), \ldots) && \text{for all } \alpha \in 2^\omega, \\ X^n &= \underbrace{X \otimes \ldots \otimes X}_{n \text{ times}} = (\underbrace{0, \ldots, 0}_{n \text{ times}}, 1, 0, 0, \ldots) && \text{for all } n \in \omega. \end{aligned}$$

A difference with the 2-adic operations occurs when looking at the bitstream $(1, 1, 1, \dots)$. In $\mathbb{A}_{mod2}$, this bitstream can only be defined with the use of the division operation. Using the ring properties of $\mathbb{A}_{mod2}$ and the nilpotency of $\oplus$, it is not difficult to see that

$$([1] \oplus X) \otimes ([1] \oplus X \oplus X^2 \oplus X^3 \dots) = [1]$$

and hence

$$(1, 1, 1, \dots) \;=\; [1] \oplus X \oplus X^2 \oplus X^3 \dots \;=\; [1] \oslash ([1] \oplus X) \;=\; \frac{[1]}{[1] \oplus X} \tag{3.9}$$

We define rational mod-2 functions analogously to rational 2-adic functions.

3.3.16. Definition. We call $\alpha \in 2^\omega$ a *polynomial mod-2 bitstream* if $\alpha = c_0 \oplus c_1 \otimes X \oplus \dots \oplus c_k \otimes X^k$ for $c_i \in \{-[1], [0], [1]\}$, and $\rho \in 2^\omega$ a *rational mod-2 bitstream* if $\rho = \alpha \oslash \beta$, with α, β polynomial mod-2 bitstreams and $\beta(0) = 1$. A function $f \colon 2^\omega \to 2^\omega$ is a *rational mod-2 function*, if $f(\sigma) = \rho_1 \oplus \rho_2 \otimes \sigma$ where ρ_1, ρ_2 are rational mod-2 bitstreams. $\lhd$

3.3.17. Remark. In the 2-adic case, there is a bijection between the eventually constant bitstreams and the polynomial 2-adic bitstreams since $\mathrm{Bin} \colon \hat{\mathbb{Q}} \to \mathbb{A}_{2adic}$ is a homomorphism of integral domains, and integer bitstreams correspond 1-1 to eventually constant bitstreams. In the mod-2 bitstream algebra, this is not the case, since $\ominus$ is identity, and hence polynomial mod-2 bitstreams are exactly the eventually constant bitstreams that end in a tail of 0's. In $\mathbb{A}_{mod2}$ eventually constant bitstreams can be defined as rational mod-2 bitstreams. From (3.9) it follows that an eventually constant bitstream $\alpha = (a_0, a_1, \dots, a_{n-1}, a_n, a_n, a_n, \dots)$, $n \geq 0$, can be defined as:

$$\alpha = [a_0] \;\oplus\; ([a_1] \otimes X) \;\oplus\; \dots \;\oplus\; ([a_{n-1}] \otimes X^{n-1}) \;\oplus\; \left([a_n] \otimes \frac{X^n}{[1] \oplus X}\right) \tag{3.10}$$

where $[a_i] \in \{[0], [1]\}$ for $0 \leq i \leq n$. $\lhd$

Just as in the 2-adic case, calculations on rational mod-2 functions is often easier when representing them in a one-fraction format.

3.3.18. Lemma. *A function $f \colon 2^\omega \to 2^\omega$ is a rational mod-2 function iff f is of the form:*

$$f(\sigma) = \frac{\delta \oplus \rho \otimes \sigma}{\pi} \tag{3.11}$$

where δ, ρ, π are eventually constant bitstreams, and $\pi(0) = 1$.

PROOF. A rational 2-adic function is of the form (3.11), since polynomial mod-2 bitstreams are eventually periodic. Conversely, using Remark 3.3.17, we observe that any function of the form (3.11) can be written as

$$f(\sigma) = \frac{\frac{\rho_1}{\pi_1} \oplus \frac{\rho_2}{\pi_2} \otimes \sigma}{\frac{\rho_3}{\pi_3}}$$

where ρ_1, ρ_2, ρ_3 are polynomial mod-2 bitstreams, $\rho_3(0) = 1$ and $\{\pi_1, \pi_2, \pi_3\} \subseteq \{[1], [1] \oplus X\}$. Using the identities of integral domains, we can multiply f with $[1] = \frac{[1]+X}{[1]+X}$ and rewrite the result into a format

$$f(\sigma) = \frac{\alpha_1 \oplus \alpha_2 \otimes \sigma}{\alpha_3}$$

where $\alpha_1, \alpha_2, \alpha_3$ are polynomial mod-2 bitstreams, and $\alpha_3(0) = 1$. QED

We now characterise the derivatives of rational mod-2 functions in an analogue of Lemma 3.3.8.

3.3.19. LEMMA. *Let $f(\sigma) = \frac{\delta \oplus \rho \otimes \sigma}{\pi}$ be a rational mod-2 function. For all $w \in 2^+$, the stream function derivative $f{\cdot}w$ is of the form*

$$(f{\cdot}w)(\sigma) = \frac{\delta_w \oplus \rho \otimes \sigma}{\pi}$$

for an eventually constant $\delta_w \in 2^\omega$ such that $\#(\delta_w) = 1$ if $K = 1$ and $\#(\delta_w) \leq K - 1$ if $K > 1$, where $K = \max\{\#(\delta), \#(\rho), \#(\pi)\}$.

PROOF. Applying the stream differential equations of Figure 3.2, we find that

$$\begin{aligned} (f{\cdot}0)(\sigma) &= \frac{([\delta(0)] \otimes \pi' \oplus \delta') \oplus \rho \otimes \sigma}{\pi} \text{ and} \\ (f{\cdot}1)(\sigma) &= \frac{([\delta(0) \oplus \rho(0)] \otimes \pi' \oplus \delta' \oplus \rho') \oplus \rho \otimes \sigma}{\pi} \end{aligned} \tag{3.12}$$

By a straightforward induction on $|w|$, we obtain that for every $w \in 2^+$, the derivative $f{\cdot}w$ is of the form $(f{\cdot}w)(\sigma) = \frac{\delta_w \oplus \rho \otimes \sigma}{\pi}$, where δ_w is an eventually constant bitstream. Moreover, it is easy to see that for any eventually constant bitstreams α and β, we have

$$\#(\alpha \oplus \beta) \leq \max\{\#(\alpha), \#(\beta)\}.$$

From this and (3.12) we get: If $K = 1$ (i.e., δ, ρ, π are constant), then $\#(\delta_w) = 1$ for all $w \in 2^+$, and if $K > 1$, then one can show by an easy induction on the length of w that for all $w \in 2^+$: $\#(\delta_w) \leq K - 1$. QED

The above lemma clearly implies that rational mod-2 functions are finite-state.

3.3.20. THEOREM. *All rational mod-2 bitstream functions are realisable.*

PROOF. Immediate from Definition 3.3.16 and Lemma 3.3.19. QED

We now use Lemma 3.3.19 to give an upper bound on the number of derivatives of rational mod-2 functions.

3.3.21. THEOREM. *Let* $f(\sigma) = \frac{\delta \oplus \rho \otimes \sigma}{\pi}$ *be a rational mod-2 function. We have:*

$$\#(f) \;\leq\; 1 + 2^{K-1}, \quad \textit{where } K = \max\{\#(\delta), \#(\rho), \#(\pi)\}.$$

PROOF. First observe that there are 2^N bitstreams α with $\#(\alpha) = N$, $N \geq 1$. We now use Lemma 3.3.19. If $K = 1$, then $\#(\delta_w) = 1$ for all $w \in 2^+$ and also $\#(\delta) = 1$, hence $\#(f) \leq 2 = 1 + 2^{K-1}$. If $K > 1$, then:

$$\#(f) \;\leq\; 1 + |\{\delta_w \mid w \in 2^+\}| \;\leq\; 1 + |\{\delta_w \mid \#(\delta_w) \leq K-1\}| \;=\; 1 + 2^{K-1}.$$

QED

3.3.22. REMARK. As described in Remark 3.3.13, Raney's results (Prop. 3.2.5) also provide us with an upper bound. A rational mod-2 function $f(\sigma) = \frac{\delta \oplus \rho \otimes \sigma}{\pi}$ is equal to the composition $f(\sigma) = (\mathrm{Div}_\pi \circ \mathrm{Add}_\delta \circ \mathrm{Mul}_\rho)(\sigma)$, where for an eventually constant bitstream $\alpha \in 2^\omega$

$$\begin{aligned} \mathrm{Mul}_\alpha(\sigma) &= \alpha \otimes \sigma, \\ \mathrm{Add}_\alpha(\sigma) &= \alpha \oplus \sigma, \\ \mathrm{Div}_\alpha(\sigma) &= \sigma \oslash \alpha \quad (\text{if } \alpha(0) = 1). \end{aligned}$$

Using an analysis similar to the one used in the proof of Lemma 3.3.19 it is possible to show that if $\#(\alpha) = n \geq 2$ then the derivatives of $\mathrm{Mul}_\alpha(\sigma)$ correspond to subsets of $\{0, 1, \ldots, n-1\}$ if $\alpha^{(n-1)} = (1, 1, 1, \ldots)$ and to subsets of $\{0, 1, \ldots, n-2\}$ if $\alpha^{(n-1)} = (0, 0, 0, \ldots)$. In the case that $\#(\alpha) = (b, b, b, \ldots)$, for $b \in 2$, we have: $\#(\mathrm{Mul}_\alpha) = 2^b$. Similarly, we find that $\#(\mathrm{Add}_\alpha) \leq \#(\alpha)$. We summarise:

$$\#(\mathrm{Mul}_\alpha) \;\leq\; \begin{cases} 2^{\#(\alpha)-2} & \text{if } \#(\alpha) \geq 2 \text{ and } \alpha^{(\#(\alpha)-1)} = (0,0,0,\ldots), \\ 2^{\#(\alpha)-1} & \text{if } \#(\alpha) \geq 2 \text{ and } \alpha^{(\#(\alpha))} = (1,1,1,\ldots), \\ 1 & \text{if } \alpha = (0,0,0,\ldots), \\ 2 & \text{if } \alpha = (1,1,1,\ldots). \end{cases}$$

$$\#(\mathrm{Add}_\alpha) \;\leq\; \#(\alpha),$$

$$\#(\mathrm{Div}_\alpha) \;=\; \#(Mul_\alpha) \quad (\text{if } \alpha(0) = 1).$$

Experiments indicate that all three bounds are tight. Applying Prop. 3.2.5 to the case where $\rho^{(\#(\rho))-1} = \pi^{(\#(\pi))-1} = [0]$, we find that

$$\#(f) \;\leq\; \#(\mathrm{Mul}_\rho) \cdot \#(\mathrm{Add}_\delta) \cdot \#(\mathrm{Div}_\pi) \;\leq\; 2^{\#(\rho)+\#(\pi)-4} \cdot \#(\delta)$$

For sufficiently large values of both $\#(\rho)$ and $\#(\pi)$ this upper bound exceeds the upper bound from Theorem 3.3.21 by an exponential factor. ◁

Often the minimal realisation will have fewer states than the upper bound given in Theorem 3.3.21, however, experimental results show that the upper bound is regularly reached. We provide an example below, which also shows that a representation of a rational mod-2 function f with polynomial δ, ρ and π does not always give an optimal upper bound on $\#(f)$.

3.3.23. Example. Consider the rational mod-2 function:

$$f(\sigma) = \frac{X^2 \oplus \frac{1 \oplus X \oplus X^2}{1 \oplus X} \otimes \sigma}{\frac{1 \oplus X^2 \oplus X^3}{1 \oplus X}}$$

That is, $\delta = X^2 = (0,0,1,0,0,0,\ldots)$,

$$\rho = \frac{1 \oplus X \oplus X^2}{1 \oplus X} = (1,0,1,1,1,\ldots),$$

$$\pi = \frac{1 \oplus X^2 \oplus X^3}{1 \oplus X} = (1,1,0,1,1,1,\ldots).$$

We have,

$$\max\{\#(\delta), \#(\rho), \#(\pi)\} = \max\{4,3,4\} = 4,$$

and by computing derivatives, we find that $\#(f) = 9 = 1 + 2^3$. The bound derived using Raney's results (see Remark 3.3.22) is $2^2 \cdot 4 \cdot 2^3 = 128$.

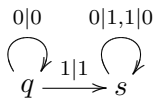
A representation of f using polynomial mod-2 bitstreams is given by:

$$\tilde{f}(\sigma) = \frac{(X^2 \oplus X^3) \oplus (1 \oplus X \oplus X^2) \otimes \sigma}{1 \oplus X^2 \oplus X^3}$$

The upper bound derived from this representation is $1 + 2^{\max\{5,4,5\}-1} = 17$. ◁

Finally, we show that, like rational 2-adic functions, rational mod-2 functions are not expressively complete for Mealy machines.

3.3.24. Example. Consider the Mealy machine $\mathbb{M}$ depicted in the following diagram:



We will show that the stream function realised by the state q is *not* a rational mod-2 function of the form $f(\sigma) = \frac{\delta \oplus \rho \otimes \sigma}{\pi}$. We first note that the function realised by the state s is $g(\sigma) = \frac{[1]}{[1]\oplus X} \oplus \sigma = \frac{[1] \oplus ([1]\oplus X)\otimes\sigma}{[1]\oplus X}$. This can easily be verified by showing that the relation $R = \{\langle s, g\rangle\}$ is a Mealy bisimulation between $\mathbb{M}$ and g in the final Mealy coalgebra.

From Lemma 3.3.19 it follows that if $\mathrm{Beh}(q)$ is rational, then it is of the form $\mathrm{Beh}(q)(\sigma) = \frac{\delta \oplus ([1]\oplus X)\otimes\sigma}{[1]\oplus X}$. Using equations (3.12) and the transition structure of $\mathbb{M}$, it follows that $f[0] = \delta(0) = 0$ and $(f{\cdot}0)(\sigma) = \frac{\delta' \oplus ([1]\oplus X)\otimes\sigma}{[1]\oplus X} = f(\sigma)$, hence $\delta' = \delta$. It follows by stream coinduction that $\delta = \delta' = [0]$. On the other hand, we also find that

$$(f{\cdot}1)(\sigma) = \frac{([1] \oplus \delta' \oplus [1]) \ \oplus \ ([1] \oplus X) \otimes \sigma}{1 \oplus X} = \frac{\delta' \ \oplus \ ([1] \oplus X) \otimes \sigma}{[1] \oplus X} = g(\sigma),$$

which implies that $\delta' = [1]$. Hence such a δ cannot exist, and we conclude that $\mathrm{Beh}(q)$ is not a rational mod-2 function.

In Example 3.3.24, we saw that $\mathrm{Beh}(q)$ can be defined as the rational 2-adic function $f(\sigma) = -\sigma$. Hence the rational 2-adic functions are not a subset of the rational mod-2 functions. ◁

3.3.25. Example. As in the 2-adic case, squaring and taking inverses in mod-2 arithmetic are not realisable. (cf. Example 3.3.15). For $f(\sigma) = \sigma \otimes \sigma$, and $n \in \mathbb{N}$, $f{\cdot}0^n = X^n \otimes \sigma \otimes \sigma$, and for $m \neq n$, clearly $f{\cdot}0^m \neq f{\cdot}0^n$. Similarly, for $g(\sigma) = [1]/([1] \oplus X \otimes \sigma)$ and $n \geq 1$, $g{\cdot}0^n = (X \otimes \sigma)/([1] \oplus X^{n+1} \otimes \sigma)$, and these derivatives are different for different n. ◁

The above example suggests that, just like in the 2-adic case, rational mod-2 functions are the most general format of realisable specifications in mod-2 arithmetic.

3.4 Implementation

In this section, we describe the implementation of our synthesis procedure. First we present our implementation of the two bitstream algebras of the previous section, together with their algebraic and coalgebraic semantics. Next we describe in some detail how we check equivalence of expressions by reducing them to normal form. Finally, we give a high-level description of the algorithm. The amount of detail is supposed to provide enough insight into the workings of the algorithm to convince the reader of its correctness. Furthermore, a good understanding of the syntax and algorithm is useful for the complexity analysis in Section 3.5.

3.4.1 Mealy coalgebra of expressions

The two bitstream algebras described in the previous section provide (different) semantics for a single arithmetic language. In our program we implement the two bitstream algebras via the algebraic data types `Bit`, `Const` (constant bitstream expressions), `Expr2Adic` (2-adic expressions), and `ExprMod2` (mod-2 expressions). The expressions of these data types are generated by the following grammars (over a single variable σ):

$$\begin{array}{ll} \texttt{Bit}: & \texttt{a} ::= \texttt{0} \mid \texttt{1} \\ \texttt{Const}: & \texttt{c} ::= [\texttt{a}] \mid \texttt{X}^{\texttt{n}}, \\ \texttt{Expr2Adic}: & \texttt{e} ::= \texttt{c} \mid \sigma \mid -\texttt{e} \mid \texttt{e} + \texttt{e} \mid \texttt{e} \times \texttt{e} \mid \texttt{e}/\texttt{e} \\ \texttt{ExprMod2}: & \texttt{e} ::= \texttt{c} \mid \sigma \mid \ominus\texttt{e} \mid \texttt{e} \oplus \texttt{e} \mid \texttt{e} \otimes \texttt{e} \mid \texttt{e} \oslash \texttt{e} \end{array}$$

where $\texttt{a} \in \texttt{Bit}$, $\texttt{c} \in \texttt{Const}$, $\texttt{n} \in \mathbb{N}$ and σ is a bitstream variable.

A 2-adic expression $\texttt{e} \in \texttt{Expr2Adic}$ is called *constant*, if the variable σ does not occur in `e`, and *polynomial*, if the division operator / does not occur in `e`. We denote the set of polynomial 2-adic expressions by `Poly2Adic` and the set of constant, polynomial 2-adic exprssions by `CPolyExpr`. Constant and polynomial mod-2 expressions are defined analogously, and we denote the set of polynomial mod-2 expressions by `PolyMod2` and the set of constant, polynomial mod-2 exprssions by `CPolyMod2`. For convenience of presentation, we let $\texttt{Expr} = \texttt{Expr2Adic} \cup \texttt{ExprMod2}$, $\texttt{PolyExpr} = \texttt{Poly2Adic} \cup \texttt{PolyMod2}$ and $\texttt{CPolyExpr} = \texttt{CPoly2Adic} \cup \texttt{CPolyMod2}$, however, `Expr`, `PolyExpr` `CPolyExpr` are not actual data types in our program.

Moreover, we use the same symbols to denote the syntactic representation of the arithmetic operations and their semantically defined bitstream operations. This should not lead to confusion, as the context will always make clear which interpretation is intended. We have chosen to include not only the constant X, but all powers of X as atomic expressions, to facilitate syntactic manipulations. As is standard, we will usually write `X` instead of $\texttt{X}^{\texttt{1}}$. We also remark that the above grammars reflect the fact that our program implements the constant bitstream expressions as a basic data type shared by `Expr2Adic` and `ExprMod2`.

We define the length, $len(\texttt{e})$, of an expression $\texttt{e} \in \texttt{Expr}$ to be the number of atomic expressions and operator symbols in `e`. That is, $[\texttt{0}], [\texttt{1}], \sigma$ and $\texttt{X}^{\texttt{n}}$, for $\texttt{n} \in \mathbb{N}$, all have length 1, and the length of a non-atomic expression is 1 plus the length of the immediate subexpressions, i.e., $len(\texttt{e}_1 + \texttt{e}_2) = 1 + len(\texttt{e}_1) + len(\texttt{e}_2)$, etc.

Our aim is to define Mealy coalgebra structures on `Expr2Adic` and `ExprMod2` such that the semantics obtained from the unique map into the final Mealy coalgebra coincides with the semantics of the operations in $\mathbb{A}_{2adic}$ and $\mathbb{A}_{mod2}$, respectively. In order to do so, we need a systematic procedure to obtain an output bit and a next-state expression from an expression `e` and a bit a. Recall

Definition 3.2.1 of the initial output and stream function derivative of a causal function $f(\sigma)$ with respect to a bit $a \in 2$:

$$f[a] = f(a{:}\sigma)(0) \quad \text{and} \quad (f{\cdot}a)(\sigma) = f(a{:}\sigma)'.$$

We observe that the initial output and derivative is obtained in two steps, which can be described informally as:

1. Substitute σ with $a{:}\sigma$, and apply f to $a{:}\sigma$ (instantiate $f(\sigma)$ with a).
2. Take initial value and stream derivative of $f(a{:}\sigma)$.

We wish to mimic this semantic definition in the syntax. The first observation we make is that given a bitstream $\alpha \in 2^\omega$ and a bit $a \in 2$, we have

$$\begin{array}{rcrcr} 0{:}\alpha & = & X \times \alpha & = & X \otimes \alpha \\ 1{:}\alpha & = & [1] + X \times \alpha & = & [1] \oplus X \otimes \alpha \end{array}$$

This means that instantiation can be carried out as a substitution operation, and leads to the following definition.

3.4.1. Definition. The function $\mathtt{inst}\colon \mathtt{Expr} \to \mathtt{Expr}^{\mathtt{Bit}}$ is defined inductively for $\mathtt{e}_1, \mathtt{e}_2 \in \mathtt{Expr}$ and $\mathtt{a} \in \mathtt{Bit}$ as follows:

$\mathrm{inst}(\mathtt{c})(\mathtt{a})$	$=$	$\mathtt{c}$,	if $\mathtt{c} \in \mathtt{Const}$,
$\mathrm{inst}(\sigma)(\mathtt{0})$	$=$	$\mathtt{X} \times \sigma$,	if $\sigma \in \mathtt{Expr2Adic}$,
$\mathrm{inst}(\sigma)(\mathtt{1})$	$=$	$[\mathtt{1}] + \mathtt{X} \times \sigma$,	if $\sigma \in \mathtt{Expr2Adic}$,
$\mathrm{inst}(\sigma)(\mathtt{0})$	$=$	$\mathtt{X} \otimes \sigma$,	if $\sigma \in \mathtt{ExprMod2}$,
$\mathrm{inst}(\sigma)(\mathtt{1})$	$=$	$[\mathtt{1}] \oplus \mathtt{X} \otimes \sigma$,	if $\sigma \in \mathtt{ExprMod2}$,
$\mathrm{inst}(\mathtt{neg}\ \mathtt{e}_1)(\mathtt{a})$	$=$	$\mathtt{neg}\ \mathrm{inst}(\mathtt{e}_1)(\mathtt{a})$,	if $\mathtt{neg} \in \{-, \ominus\}$,
$\mathrm{inst}(\mathtt{e}_1\ \mathtt{op}\ \mathtt{e}_2)(\mathtt{a})$	$=$	$\mathrm{inst}(\mathtt{e}_1)(\mathtt{a})\ \mathtt{op}\ \mathrm{inst}(\mathtt{e}_2)(\mathtt{a})$,	if $\mathtt{op} \in \{+, \times, /, \oplus, \otimes, \oslash\}$.

The set $\mathtt{IExpr} = \{\mathrm{inst}(\mathtt{e})(\mathtt{a}) \mid \mathtt{e} \in \mathtt{Expr}, a \in \mathtt{Bit}\}$ will be called the set of *instantiated expressions*, and we let $\mathtt{IExpr2Adic} = \mathtt{IExpr} \cap \mathtt{Expr2Adic}$ and $\mathtt{IExprMod2} = \mathtt{IExpr} \cap \mathtt{ExprMod2}$. ◁

Clearly, for all $\mathtt{e} \in \mathtt{Expr2Adic}$ and any $\mathtt{a} \in \mathtt{Bit}$, we have $\mathrm{inst}(\mathtt{e}, \mathtt{a}) \in \mathtt{Expr2Adic}$. Similarly, for $\mathtt{e} \in \mathtt{ExprMod2}$ and $\mathtt{a} \in \mathtt{Bit}$, we have $\mathrm{inst}(\mathtt{e}, \mathtt{a}) \in \mathtt{ExprMod2}$. Hence inst induces maps $\mathrm{inst}_{2adic}\colon \mathtt{Expr2Adic} \to \mathtt{Expr2Adic}^{\mathtt{Bit}}$ and $\mathrm{inst}_{mod2}\colon \mathtt{ExprMod2} \to \mathtt{ExprMod2}^{\mathtt{Bit}}$.

To carry out step 2 of obtaining a Mealy coalgebra of expressions, we will define a bitstream automaton structure on $\mathtt{IExpr}$, in other words, we will define initial value and stream derivatives of instantiated expressions. For the constant bitstream expressions, it is clear how to do this. To deal with variable expressions, we note that in all instantiated expressions $\mathtt{e} \in \mathtt{IExpr}$, the variable σ

occurs only within subexpressions of the form $\mathtt{X} \times \sigma$ or $\mathtt{X} \otimes \sigma$, so given the fact that for all bitstreams $\alpha \in 2^\omega$, $X \times \alpha = X \otimes \alpha = 0 : \alpha$, it is now also clear how to define initial value and derivative of $\mathtt{X} \times \sigma$ and $\mathtt{X} \otimes \sigma$. For the arithmetic operations we simply use their defining stream differential equations in Figure 3.1 (p. 30) and Figure 3.2 (p. 39). However, due to the condition that we can only divide with bitstreams with initial value 1, some expressions will not have a well-defined stream behaviour.

3.4.2. Definition. Let `xor` and & be operation symbols that denote the (semantic) operations of exclusive-or (addition mod-2) and Boolean conjunction on bits. Let $\mathbf{1} = \{\star\}$ be the singleton set containing the *undefined value* $\star$. The map $\langle h, t\rangle \colon \mathtt{IExpr} \to \mathbf{1} + \mathtt{Bit} \times \mathtt{Expr}$ is defined in the following table:

$\mathtt{e}$	$h(\mathtt{e})$	$t(\mathtt{e})$	condition
$[0]$	0	$[0]$	
$[1]$	1	$[0]$	
$\mathtt{X}^0$	1	$[0]$	
$\mathtt{X}^\mathtt{n}$	0	$\mathtt{X}^{\mathtt{n}-1}$	if $\mathtt{n} \geq 1$
$\mathtt{X} \times \sigma$	0	σ	
$\mathtt{X} \otimes \sigma$	0	σ	
$-\mathtt{e}_1$	$h(\mathtt{e}_1)$	$-(t(\mathtt{e}_1) + [h(\mathtt{e}_1)])$	
$\mathtt{e}_1 + \mathtt{e}_2$	$h(\mathtt{e}_1)\,\mathtt{xor}\,h(\mathtt{e}_2)$	$t(\mathtt{e}_1) + (t(\mathtt{e}_2) + [h(\mathtt{e}_1) \,\&\, h(\mathtt{e}_2)])$	
$\mathtt{e}_1 \times \mathtt{e}_2$	$h(\mathtt{e}_1) \,\&\, h(\mathtt{e}_2)$	$t(\mathtt{e}_1) \times \mathtt{e}_2 + [h(\mathtt{e}_1)] \times t(\mathtt{e}_2)$	
$\mathtt{e}_1/\mathtt{e}_2$	$h(\mathtt{e}_1)$	$(t(\mathtt{e}_1) + (-[h(\mathtt{e}_1)] \times t(\mathtt{e}_2)))/\mathtt{e}_2$	if $h(\mathtt{e}_1) = 1$
$\ominus\mathtt{e}_1$	$h(\mathtt{e}_1)$	$\ominus\, t(\mathtt{e}_1)$	
$\mathtt{e}_1 \oplus \mathtt{e}_2$	$h(\mathtt{e}_1)\,\mathtt{xor}\,h(\mathtt{e}_2)$	$t(\mathtt{e}_1) \oplus t(\mathtt{e}_2)$	
$\mathtt{e}_1 \otimes \mathtt{e}_2$	$h(\mathtt{e}_1) \,\&\, h(\mathtt{e}_2)$	$t(\mathtt{e}_1) \otimes \mathtt{e}_2 \oplus [h(\mathtt{e}_1)] \otimes t(\mathtt{e}_2)$	
$\mathtt{e}_1 \oslash \mathtt{e}_2$	$h(\mathtt{e}_1)$	$(t(\mathtt{e}_1) \oplus [h(\mathtt{e}_1)] \otimes t(\mathtt{e}_2)) \oslash \mathtt{e}_2$	if $h(\mathtt{e}_1) = 1$

If the condition for the division operations is not satisfied, or $\mathtt{e}$ contains a subexpression $\mathtt{f}$ for which $\langle h(\mathtt{f}), t(\mathtt{f})\rangle = \star$, then $\langle h(\mathtt{e}), t(\mathtt{e})\rangle = \star$. ◁

Again, we note that h and t restrict to maps $h_{2adic} \colon \mathtt{IExpr2Adic} \to \mathbf{1} + \mathtt{Bit}$ and $t_{2adic} \colon \mathtt{IExpr2Adic} \to \mathbf{1} + \mathtt{Expr2Adic}$; and similarly for `ExprMod2`.

A partial Mealy coalgebra structure on `Expr` is now obtained by composing instantiation with stream behaviour,

$$\mathtt{Expr} \xrightarrow{\text{inst}} \mathtt{IExpr}^{\mathtt{Bit}} \xrightarrow{\langle h,t\rangle^{\mathtt{Bit}}} (\mathbf{1} + \mathtt{Bit} \times \mathtt{Expr})^{\mathtt{Bit}}$$

and partial Mealy coalgebra structures on `Expr2Adic` and `ExprMod2` are obtained by restriction.

3.4.3. Definition (Mealy coalgebra of expressions). Let the mapping $\eta_{2adic}\colon \mathtt{Expr2Adic} \to (\mathbf{1} + \mathtt{Bit} \times \mathtt{Expr2Adic})^{\mathtt{Bit}}$ be defined for all $\mathtt{e} \in \mathtt{Expr2Adic}$ and $\mathtt{a} \in \mathtt{Bit}$ by:

$$\eta_{2adic}(\mathtt{e})(\mathtt{a}) \quad = \quad \langle h_{2adic}(\mathrm{inst}_{2adic}(\mathtt{e})(\mathtt{a})),\, t_{2adic}(\mathrm{inst}_{2adic}(\mathtt{e})(\mathtt{a}))\rangle.$$

The map $\eta_{mod2}\colon \mathtt{ExprMod2} \to (\mathbf{1} + \mathtt{Bit} \times \mathtt{ExprMod2})^{\mathtt{Bit}}$ is defined for all $\mathtt{e} \in \mathtt{ExprMod2}$ and $\mathtt{a} \in \mathtt{Bit}$ by:

$$\eta_{mod2}(\mathtt{e})(\mathtt{a}) \quad = \quad \langle h_{mod2}(\mathrm{inst}_{mod2}(\mathtt{e})(\mathtt{a})),\, t_{mod2}(\mathrm{inst}_{mod2}(\mathtt{e})(\mathtt{a}))\rangle. \qquad \triangleleft$$

3.4.4. Remark. The partiality of the structures defined above seems undesirable. It could be avoided if we restrict `Expr2Adic` to contain only fractions $\mathtt{e}_1/\mathtt{e}_2$ where $h(\mathtt{e}_2) = 1$. This restriction on the syntax is, however, difficult to implement. So for now, we accept that not all expressions have a well-defined Mealy behaviour. $\triangleleft$

An expression $\mathtt{e} \in \mathtt{Expr}$ is called a *Mealy expression*, if the subcoalgebra $\langle \mathtt{e} \rangle$ generated by $\mathtt{e}$ does not contain the undefined value $\star$. This condition is equivalent with $\langle \mathtt{e} \rangle$ being isomorphic to a (total) Mealy coalgebra, and hence the final map $\mathrm{Beh}\colon \langle \mathtt{e} \rangle \to \mathbf{\Gamma}$ provides $\mathtt{e}$ with causal bitstream function semantics. We say that a Mealy expression $\mathtt{e}$ is a *specification* of the causal bitstream function $\mathrm{Beh}(\mathtt{e})$. An expression $\mathtt{e} \in \mathtt{Expr2Adic}$ is called a *rational 2-adic function specification*, if $\mathtt{e} = (d + m \times \sigma)/n$ where $\mathtt{d}, \mathtt{m}, \mathtt{n} \in \mathtt{Poly2Adic}$. Similarly, $\mathtt{e} \in \mathtt{ExprMod2}$ is a *rational mod-2 function specification*, if $\mathtt{e} = (d \oplus m \otimes \sigma) \oslash n$ where $\mathtt{d}, \mathtt{m}, \mathtt{n} \in \mathtt{PolyMod2}$. From the definitions of rational 2-adic and mod-2 functions (Definition 3.3.3 and 3.3.16) it should be clear that if $\mathtt{e}$ is a rational 2-adic (mod-2) function specification then $\mathrm{Beh}(\mathtt{e})$ is a rational 2-adic (mod-2) function. Conversely, all rational 2-adic (mod-2) functions are the behaviour of some rational 2-adic (mod-2) function specification.

The bitstream algebras $\mathbb{A}_{2adic}$ and $\mathbb{A}_{mod2}$ provide semantics for expressions from `Expr2Adic` and `ExprMod2` in the obvious way. For a Mealy expression $\mathtt{e} \in \mathtt{Expr2Adic}$, we denote by $\mathtt{e}(\alpha)^{\mathbb{A}_{2adic}}$ the bitstream obtained by evaluating $\mathtt{e}$ in $\mathbb{A}_{2adic}$ with α assigned to σ. For two Mealy expressions $\mathtt{e}_1, \mathtt{e}_2 \in \mathtt{Expr2Adic}$, we write $\mathbb{A}_{2adic} \models \mathtt{e}_1 \equiv \mathtt{e}_2$ if for all $\alpha \in 2^\omega$: $\mathtt{e}_1(\alpha)^{\mathbb{A}_{2adic}} = \mathtt{e}_2(\alpha)^{\mathbb{A}_{2adic}}$, and say that $\mathtt{e}_1$ and $\mathtt{e}_2$ are *algebraically equivalent*. Evaluation and equivalence is defined analogously for Mealy expressions in `ExprMod2`. Intuitively, it is clear that two Mealy expressions are algebraically equivalent if and only if they specify the same bitstream function, i.e., they have the same Mealy behaviour. We denote behavioural equivalence in Mealy coalgebras by $\sim_{\mathcal{M}}$.

3.4.5. Proposition. *For all Mealy expressions* $\mathtt{e}_1, \mathtt{e}_2 \in \mathtt{Expr2Adic}$*:*

$$\mathbb{A}_{2adic} \models \mathtt{e}_1 \equiv \mathtt{e}_2 \quad \textit{iff} \quad \mathtt{e}_1 \sim_{\mathcal{M}} \mathtt{e}_2.$$

For all Mealy expressions $\mathtt{e}_1, \mathtt{e}_2 \in \mathtt{ExprMod2}$*:*

$$\mathbb{A}_{mod2} \models \mathtt{e}_1 \equiv \mathtt{e}_2 \quad \textit{iff} \quad \mathtt{e}_1 \sim_{\mathcal{M}} \mathtt{e}_2.$$

Proof. We only show the case for `Expr2Adic`. We have defined the structure η_{2adic} such that for all Mealy expressions $\mathtt{e} \in \mathtt{Expr2Adic}$, and all $\alpha \in 2^\omega$:

$$\mathrm{Beh}(\mathtt{e})(\alpha) = \mathtt{e}(\alpha)^{\mathbb{A}_{2adic}}. \tag{3.13}$$

This can be verified using stream coinduction. It now follows that:

$$\begin{array}{lll} \mathbb{A}_{2adic} \models \mathtt{e}_1 \equiv \mathtt{e}_2 & \text{iff} & \text{for all } \alpha \in 2^\omega : \mathtt{e}_1(\alpha)^{\mathbb{A}_{2adic}} = \mathtt{e}_2(\alpha)^{\mathbb{A}_{2adic}} \\ & \text{iff} & \text{for all } \alpha \in 2^\omega : \mathrm{Beh}(\mathtt{e}_1)(\alpha) = \mathrm{Beh}(\mathtt{e}_2)(\alpha) \\ & \text{iff} & \mathrm{Beh}(\mathtt{e}_1) = \mathrm{Beh}(\mathtt{e}_2) \\ & \text{iff} & \mathtt{e}_1 \sim_{\mathcal{M}} \mathtt{e}_2. \end{array}$$ QED

The above proposition may seem trivial, since we have simply defined the Mealy coalgebra of expressions such that it holds. However, Proposition 3.4.5 is essential for our synthesis algorithm, since it explains and justifies our use of equational logic and rewriting in deciding equivalence of expressions. This is the subject of the next subsection.

3.4.6. Example. Consider the 2-adic expression $\mathtt{e} = \frac{X^2 \times \sigma}{[1]+X}$. We compute the transition of $\mathtt{e}$ on the bit `1`. The instantiation of $\mathtt{e}$ with `1`, is

$$\mathrm{inst}(\mathtt{e})(\mathtt{1}) = \frac{X^2 \times ([1] + X \times \sigma)}{[1] + X},$$

and we find that

$$h(\mathrm{inst}(\mathtt{e})(\mathtt{1})) \;=\; h(\mathtt{X}^2) \,\&\, h([\mathtt{1}] + \mathtt{X} \times \sigma) \;=\; \mathtt{0} \,\&\, \mathtt{1} \;=\; \mathtt{0} \text{ and}$$

$$\begin{aligned} t(\mathrm{inst}(\mathtt{e})(\mathtt{1})) \;&=\; t\left(\frac{\mathtt{X}^2 \times ([\mathtt{1}] + \mathtt{X} \times \sigma)}{[\mathtt{1}] + \mathtt{X}}\right) \\ &=\; \frac{t(\mathtt{X}^2 \times ([\mathtt{1}] + \mathtt{X} \times \sigma)) - h(\mathtt{X}^2 \times ([\mathtt{1}] + \mathtt{X} \times \sigma)) \times t([\mathtt{1}] + \mathtt{X})}{[\mathtt{1}] + \mathtt{X}} \\ &=\; \frac{t(\mathtt{X}^2) \times ([\mathtt{1}] + \mathtt{X} \times \sigma) + h(\mathtt{X}^2) \times t([\mathtt{1}] + \mathtt{X} \times \sigma) - [\mathtt{0}] \times t([\mathtt{1}] + \mathtt{X})}{[\mathtt{1}] + \mathtt{X}} \\ &=\; \frac{(\mathtt{X} \times ([\mathtt{1}] + \mathtt{X} \times \sigma) + [\mathtt{0}] \times ([\mathtt{0}] + \sigma + [\mathtt{0}])) - [\mathtt{0}] \times ([\mathtt{0}] + [\mathtt{1}] + [\mathtt{0}])}{[\mathtt{1}] + \mathtt{X}} \end{aligned}$$

This expression can be simplified using the identities of the 2-adic bitstream algebra to yield the equivalent expression $\frac{\mathtt{X}+\mathtt{X}^2\times\sigma}{[\mathtt{1}]+\mathtt{X}}$. The computation of such a reduced form is explained in the next section. ◁

3.4.2 Equivalence of expressions

In order to guarantee termination of our synthesis procedure, and minimality of the constructed automaton, it is crucial that we can effectively decide whether two Mealy expressions specify the same function. Proposition 3.4.5 tells us that this amounts to deciding algebraic equivalence in the relevant bitstream algebra. In this subsection we describe what is essentially a terminating and confluent rewrite system for polynomial expressions. Consequently, we obtain a procedure for deciding equivalence of arbitrary expressions by reduction to rational normal form. We do not describe the rewrite rules in full detail, since they correspond to the well-known identities of integral domains together with the identities specific to each bitstream algebra. We also remark that the reduction to normal form is not implemented as a pure rewrite system. Some of the syntactic manipulations are performed using specialised data structures.

Most of the definitions and results of this subsection apply to both `Expr2Adic` and `ExprMod2`, so in order to avoid too many repetitions or analogous statements, we often phrase definitions and results simply for "expressions". Such a formulation should be read as two statements, one about `Expr2Adic` and one about `ExprMod2`, with the obvious adaptations. In such "generic" formulations we use the symbols $-, +, \times, /$ in a generic way, but they should be read as the 2-adic operations, respectively the mod-2 operations, as appropriate. We stress that we use the symbol $=$ to denote syntactic equality on expressions only, hence, for example, $[0] + [1] \sim_{\mathcal{M}} [1]$, but $[0] + [1] \neq [1]$.

Since the two bitstream algebras are integral domains, they satisfy the following properties, known from the rational numbers.

1. For any expression $\mathtt{e}$ there are polynomial expressions $\mathtt{p}$ and $\mathtt{q}$ such that $\mathtt{e} \equiv \mathtt{p}/\mathtt{q}$.
2. Let $\mathtt{e}_1 = \mathtt{p}_1/\mathtt{q}_1$ and $\mathtt{e}_2 = \mathtt{p}_2/\mathtt{q}_2$ be expressions. We have: $\mathtt{e}_1 \equiv \mathtt{e}_2$ if and only if $\mathtt{p}_1 \times \mathtt{q}_2 \equiv \mathtt{p}_2 \times \mathtt{q}_1$.

We first show that in step 1, $\mathtt{p}$ and $\mathtt{q}$ can be effectively computed.

3.4.7. Lemma. *There exists a function* $\mathrm{mkRat} \colon \mathtt{Expr} \to \mathtt{Expr}$*, which for any* $\mathtt{e} \in \mathtt{Expr}$ *returns a fraction of polynomial expressions* $\mathrm{mkRat}(\mathtt{e}) = \mathtt{p}/\mathtt{q}$ *such that* $\mathrm{mkRat}(\mathtt{e}) \equiv \mathtt{e}$.

Proof. Define mkRat as follows:

$$
\begin{array}{lcll}
\mathrm{mkRat}(\mathtt{e}) & = & \frac{\mathtt{e}}{[1]} & \text{if } \mathtt{e} \in \mathtt{Const} \cup \{\sigma\}; \\
\mathrm{mkRat}(-\mathtt{e}) & = & \frac{-\mathtt{p}}{\mathtt{q}} & \text{where } \frac{\mathtt{p}}{\mathtt{q}} = \mathrm{mkRat}(\mathtt{e}); \\
\mathrm{mkRat}(\mathtt{e}_1 + \mathtt{e}_2) & = & \frac{p_1 + p_2}{q_1} & \text{where } \frac{\mathtt{p}_1}{\mathtt{q}_1} = \mathrm{mkRat}(\mathtt{e}_1), \frac{\mathtt{p}_2}{\mathtt{q}_2} = \mathrm{mkRat}(\mathtt{e}_2) \\
 & & & \text{and } \mathtt{q}_1 = \mathtt{q}_2;
\end{array}
$$

$$\begin{array}{lcll}
\mathrm{mkRat}(\mathtt{e_1}+\mathtt{e_2}) & = & \frac{p_1\times q_2+p_2\times q_1}{q_1\times q_2} & \text{where } \frac{\mathtt{p_1}}{\mathtt{q_1}}=\mathrm{mkRat}(\mathtt{e_1}),\frac{\mathtt{p_2}}{\mathtt{q_2}}=\mathrm{mkRat}(\mathtt{e_2}) \\
 & & & \text{and } \mathtt{q_1}\neq\mathtt{q_2}; \\
\mathrm{mkRat}(\mathtt{e_1}\times\mathtt{e_2}) & = & \frac{p_1\times p_2}{q_1\times q_2} & \text{where } \frac{\mathtt{p_1}}{\mathtt{q_1}}=\mathrm{mkRat}(\mathtt{e_1}),\frac{\mathtt{p_2}}{\mathtt{q_2}}=\mathrm{mkRat}(\mathtt{e_2}); \\
\mathrm{mkRat}(\frac{\mathtt{e_1}}{\mathtt{e_2}}) & = & \frac{p_1\times q_2}{q_1\times p_2} & \text{where } \frac{\mathtt{p_1}}{\mathtt{q_1}}=\mathrm{mkRat}(\mathtt{e_1}),\frac{\mathtt{p_2}}{\mathtt{q_2}}=\mathrm{mkRat}(\mathtt{e_2}).
\end{array}$$

QED

We now show that we can reduce polynomial expressions to a unique normal form. This reduction proceeds in two steps. The first step is common to both `Expr2Adic` and `ExprMod2` as it uses only the ring identities common to both bitstream algebras:

3.4.8. Definition (Ring normal form). Let $\mathtt{p}$ be a polynomial `Expr2Adic`-expression. The *ring normal form of* $\mathtt{p}$ is the expression

$$\mathtt{c_0}\times\sigma^{n_0}+(\mathtt{c_1}\times\sigma^{n_1}+(\ldots+\mathtt{c_k}\times\sigma^{n_k})\ldots)$$

obtained from $\mathtt{p}$ by applying the ring identities (i.e., distribute $\times$ over $+$, group together σ-factors to obtain the constant expressions $\mathtt{c_i}$, order on the σ exponents $n_0,\ldots,n_k$, and associate $+$ to the right). We note here that σ^n is meant to be read as the n-fold product of σ with itself. The ring normal form of polynomial `ExprMod2`-expressions is defined analogously. ◁

3.4.9. Example. Let $\mathtt{p}=([\mathtt{1}]+\sigma)\times(\mathtt{X}+-\sigma)$. We obtain the ring normal form of $\mathtt{p}$ as follows:

$$\begin{array}{rcl}
([\mathtt{1}]+\sigma)\times(\mathtt{X}+-\sigma) & = & [\mathtt{1}]\times\mathtt{X}\ +\ [\mathtt{1}]\times-\sigma\ +\ \sigma\times X+\sigma\times-\sigma \\
 & = & ([\mathtt{1}]\times\mathtt{X})\ +\ ((-[\mathtt{1}]+\mathtt{X})\times\sigma\ +\ -[\mathtt{1}]\times(\sigma\times\sigma))
\end{array}$$

◁

The polynomial normal form is obtained from the ring normal form through a further reduction of the constant, polynomial coefficients $\mathtt{c_0},\ldots,\mathtt{c_k}$. This reduction is specific to each data type, and we only describe it informally. The following definition will be useful when describing this reduction for the 2-adic case, and also later on in the complexity analysis.

3.4.10. Definition. We define the map val: `Poly2Adic` $\to\mathbb{Z}$ by:

$$\begin{array}{lcllcl}
\mathrm{val}([\mathtt{0}]) & = & 0, & \mathrm{val}(-\mathtt{c}) & = & -\mathrm{val}(\mathtt{c}), \\
\mathrm{val}([\mathtt{1}]) & = & 1, & \mathrm{val}(\mathtt{c_1}+\mathtt{c_2}) & = & \mathrm{val}(\mathtt{c_1})+\mathrm{val}(\mathtt{c_2}), \\
\mathrm{val}(\mathtt{X^n}) & = & 2^n, & \mathrm{val}(\mathtt{c_1}\times\mathtt{c_2}) & = & \mathrm{val}(\mathtt{c_1})\cdot\mathrm{val}(\mathtt{c_2}).
\end{array}$$

◁

3.4.11. Definition (Constant polynomial normal form). Given a constant polynomial expression $\mathtt{c}$, we define $\mathrm{cpnf}(\mathtt{c})$ as follows, depending on the type of $\mathtt{c}$.

$\mathtt{c} \in \mathtt{CPoly2Adic}$**:** If $\mathrm{val}(\mathtt{c}) \geq 0$, then $\mathrm{cpnf}(\mathtt{c})$ is the expression which corresponds to the binary expansion of $\mathrm{val}(\mathtt{c})$ (with $\mathtt{X}$ playing the role of 2). If $\mathrm{val}(\mathtt{c}) < 0$, then $\mathrm{cpnf}(\mathtt{c}) = -\mathrm{cpnf}(-\mathtt{c})$. For example, $\mathrm{val}(\mathtt{X}^3 + \mathtt{X}^2 + -[\mathtt{1}]) = 11$ and hence $\mathrm{cpnf}(\mathtt{X}^3 + \mathtt{X}^2 + -[\mathtt{1}]) = [\mathtt{1}] + (\mathtt{X} + \mathtt{X}^3)$. Similarly, we have that $\mathrm{cpnf}(-\mathtt{X}^3 + -\mathtt{X}^2 + [\mathtt{1}]) = -([\mathtt{1}] + (\mathtt{X} + \mathtt{X}^3))$. We point out that $\mathrm{cpnf}(\mathtt{c})$ is obtained through a purely symbolic manipulation of expressions which makes only indirect use of Haskell's integer arithmetic.

$\mathtt{c} \in \mathtt{CPolyMod2}$**:** Any constant, polynomial mod-2 expression $\mathtt{c}$ can be rewritten to a sum of signed powers of $\mathtt{X}$, for example, $\mathtt{X}^2 \oplus \mathtt{X}^1 \oplus \mathtt{X}^0 \oplus \ominus\mathtt{X}^3 \oplus \mathtt{X}^2$, (by applying distributivity and the ring identity $\mathtt{X}^{\mathtt{n}} \times \mathtt{X}^{\mathtt{m}} = \mathtt{X}^{\mathtt{n}+\mathtt{m}}$). Such an expression is further reduced by applying the identities $\alpha \oplus \alpha = [0]$ and $\ominus\alpha = \alpha$, and finally ordering on exponents. The resulting normal form consists of a sum of unique powers $\mathtt{X}^{\mathtt{n}}$ ordered ascendingly on n. For example, $\mathrm{cpnf}(\mathtt{X}^2 \oplus \mathtt{X}^1 \oplus \mathtt{X}^0 \oplus \ominus\mathtt{X}^3 \oplus \mathtt{X}^2) = [\mathtt{1}] \oplus (\mathtt{X}^1 \oplus \mathtt{X}^3)$.

If $\mathrm{cpnf}(\mathtt{c}) = \mathtt{c}$, then we say $\mathtt{c}$ is in (constant, polynomial) normal form. ◁

3.4.12. Definition (Polynomial normal form). Let $\mathtt{p}$ be a polynomial expression, with ring normal form: $\mathtt{c}_0 \times \sigma^{\mathtt{n}_0} + (\mathtt{c}_1 \times \sigma^{\mathtt{n}_1} + \ldots + \mathtt{c}_{\mathtt{k}} \times \sigma^{\mathtt{n}_{\mathtt{k}}})$. The polynomial normal form $\mathrm{pnf}(\mathtt{p})$ of $\mathtt{p}$ is obtained from

$$\mathrm{cpnf}(\mathtt{c}_0) \times \sigma^{\mathtt{n}_0} + (\mathrm{cpnf}(\mathtt{c}_1) \times \sigma^{\mathtt{n}_1} + \ldots + \mathrm{cpnf}(\mathtt{c}_{\mathtt{k}}) \times \sigma^{\mathtt{n}_{\mathtt{k}}})$$

by a further reduction using the the ring identities, such that terms of the form $[\mathtt{0}] \times \sigma^{\mathtt{n}_{\mathtt{i}}}$ are removed, and terms of the form $[\mathtt{1}] \times \sigma^{\mathtt{n}_{\mathtt{i}}}$ are reduced to $\sigma^{\mathtt{n}_{\mathtt{i}}}$.

We say that $\mathtt{p}$ is in (polynomial) normal form, if $\mathrm{pnf}(\mathtt{p}) = \mathtt{p}$. ◁

3.4.13. Lemma.

1. *If* $\mathtt{c}_1$ *and* $\mathtt{c}_2$ *are constant, polynomial expressions (of the same type), then:* $\mathtt{c}_1 \equiv \mathtt{c}_2$ *iff* $\mathrm{cpnf}(\mathtt{c}_1) = \mathrm{cpnf}(\mathtt{c}_2)$.
2. *If* $\mathtt{p}_1$ *and* $\mathtt{p}_2$ *are polynomial expressions (of the same type), then:* $\mathtt{p}_1 \equiv \mathtt{p}_2$ *iff* $\mathrm{pnf}(\mathtt{p}_1) = \mathrm{pnf}(\mathtt{p}_2)$.

Proof. Item 2 follows from item 1. Item 1 for $\mathtt{c}_1, \mathtt{c}_2 \in \mathtt{CPoly2Adic}$ follows essentially from the uniqueness of binary expansions. Item 1 for $\mathtt{c}_1, \mathtt{c}_2 \in \mathtt{CPolyMod2}$ follows from the easy observation that for any $\mathtt{c} \in \mathtt{CPolyMod2}$, $\mathrm{cpnf}(\mathtt{c})$ satisfies the following condition: $\mathit{len}(\mathrm{cpnf}(\mathtt{c})) = \min\{\mathit{len}(\mathtt{c}') \mid \mathtt{c} \sim_{\mathcal{M}} \mathtt{c}'\}$. Such a "minimal" expression must be a sum of unique powers of $\mathtt{X}$, and by ordering the sum on the exponents of $\mathtt{X}$, and making $\oplus$ associate right, we obtain a unique one, which is $\mathrm{cpnf}(\mathtt{c})$. QED

3.4.14. Definition (Rational normal form). Let $\mathtt{e}$ be an expression. The *rational normal form* of $\mathtt{e}$, denoted $\mathrm{rnf}(\mathtt{e})$, is obtained as follows:

1. Rewrite $\mathtt{e}$ into a fraction of polynomial expressions: $\mathtt{p}/\mathtt{q} = \mathrm{mkRat}(\mathtt{e})$.
2. Reduce $\mathtt{p}$ and $\mathtt{q}$ to polynomial normal form.

We say that $\mathtt{e}$ is in rational normal form, if $\mathrm{rnf}(\mathtt{e}) = \mathtt{e}$. ◁

We point out that if $\mathtt{e} = \mathtt{p}/\mathtt{q}$ is in rational normal form, the polynomial expressions $\mathtt{p}$ and $\mathtt{q}$ are not necessarily coprime. For example, $(\mathtt{X}^2 \times \sigma)/\mathtt{X}^2$ is in rational normal form.

3.4.15. Example. Consider the 2-adic expression $\mathtt{e}_1 = \frac{[1]}{[1]+\mathtt{X}} + \sigma + [1]$. After computing $\mathrm{mkRat}(\mathtt{e}_1) = \mathtt{p}/\mathtt{q}$ (step 1), and reducing $\mathtt{p}$ and $\mathtt{q}$ to ring normal form, we have the expression $\frac{([1]+[1]+\mathtt{X})+([1]+\mathtt{X})\times\sigma}{[1]+\mathtt{X}}$. After reducing the constant coefficients, we obtain the rational normal form: $\mathrm{rnf}(\mathtt{e}_2) = \frac{\mathtt{X}^2+([1]+\mathtt{X})\times\sigma}{[1]+\mathtt{X}}$.

Now consider the mod-2 expression $\mathtt{e}_2 = \frac{[1]}{[1]\oplus\mathtt{X}} \oplus \sigma \oplus [1]$. First $\mathtt{e}_2$ is rewritten to the fraction of polynomial expressions in ring normal form $\frac{([1]\oplus[1]\oplus\mathtt{X})\oplus([1]\oplus\mathtt{X})\otimes\sigma}{[1]\oplus\mathtt{X}}$. The coefficients are now reduced to yield the rational normal form: $\mathrm{rnf}(\mathtt{e}_2) = \frac{\mathtt{X}\oplus([1]\oplus\mathtt{X})\otimes\sigma}{[1]\oplus\mathtt{X}}$. ◁

3.4.16. Proposition. *Let* $\mathtt{e}_1 = \mathtt{p}_1/\mathtt{q}_1$ *and* $\mathtt{e}_2 = \mathtt{p}_2/\mathtt{q}_2$ *be expressions in rational normal form with* $\mathtt{q}_1 \neq [0]$ *and* $\mathtt{q}_2 \neq [0]$.

$$\mathtt{e}_1 \sim_{\mathcal{M}} \mathtt{e}_2 \quad \textit{if and only if} \quad \mathrm{pnf}(\mathtt{p}_1 \times \mathtt{q}_2) = \mathrm{pnf}(\mathtt{p}_2 \times \mathtt{q}_1).$$

Proof. Follows from Proposition 3.4.5, Lemma 3.4.13, and the fact the two bitstream algebras are integral domains. QED

3.4.3 Algorithm

Our synthesis algorithm is a standard, symbolic fixpoint computation. Starting from the initial specification $\mathtt{e}$, we compute for each bit $a \in 2$, the transitions corresponding to input a (using Definition 3.4.3), and iterate this for the derivatives of $\mathtt{e}$ until no new transitions are found, i.e., a fixpoint has been reached. In order to detect the fixpoint, and ensure we only compute each transition once, we keep track of the new states that are added in each iteration, and to accomodate the equivalence check, we reduce all state expressions to rational normal form. More precisely, during the construction, we maintain sets of the states and transitions found in previous iterations (*Prev* and *Trans*), and a set of the states that were newly added in the previous iteration (*New*). *Prev* and *Trans*

are initialised as the empty set and *New* is initialised as the singleton $\{\mathrm{rnf}(\mathtt{e})\}$, the rational normal form of the initial specification. At the end of iteration d, *Trans* is a set of transitions representing $\langle \mathrm{Beh}(\mathtt{e}) \rangle$ up to depth d. The algorithm is described in Table 3.1.

input: $\mathtt{e}$.
Prev $:= \emptyset$; *Trans* $:= \emptyset$; *New* $:= \{\mathrm{rnf}(\mathtt{e})\}$;
while *New* $\neq \emptyset$

1. *NewTrans* $:= \emptyset$;
2. for each $\mathtt{d}$ in *New*,
   (a) compute transitions $\mathtt{d} \xrightarrow{0|a_0} \mathtt{d_0}$ and $\mathtt{d} \xrightarrow{1|a_1} \mathtt{d_1}$;
   (b) compute $\mathrm{rnf}(\mathtt{d_0})$ and $\mathrm{rnf}(\mathtt{d_1})$;
   (c) add $\mathtt{d} \xrightarrow{0|a_0} \mathrm{rnf}(\mathtt{d_0})$ and $\mathtt{d} \xrightarrow{1|a_1} \mathrm{rnf}(\mathtt{d_1})$ to *NewTrans*;
3. *Prev* $:=$ *Prev* $\cup$ *New*; *New* $:= \emptyset$; *UpdNewTrans* $:= \emptyset$;
4. for each $\mathtt{d} \xrightarrow{a|b} \mathtt{d_a} \in$ *NewTrans*,
   if $\mathtt{d_a} \sim_{\mathcal{M}} \mathtt{f}$ for some $\mathtt{f} \in$ *Prev*
   then: add $\mathtt{d} \xrightarrow{a|b} \mathtt{f}$ to *UpdNewTrans*;
   else: add $\mathtt{d} \xrightarrow{a|b} \mathtt{d_a}$ to *UpdNewTrans*, and add $\mathtt{d_a}$ to *New*;
5. *Trans* $:=$ *Trans* $\cup$ *UpdNewTrans*;
6. remove duplicates from *New* with respect to $\sim_{\mathcal{M}}$;

end while;
return *Trans*;

Table 3.1: Algorithm for the construction of a Mealy machine $\langle \mathrm{Beh}(\mathtt{e}) \rangle$ from a rational specification $\mathtt{e}$.

We note that if the input $\mathtt{e}$ has no Mealy behaviour, then the construction (and our program) will get stuck at some point, and abort, and if $\mathtt{e}$ has infinite-state behaviour, then the process will not terminate. In order to deal with the latter problem, our program provides the option of specifying the maximum automaton depth. We illustrate the construction by means of an example.

3.4.17. Example. Let $\mathtt{e} = \mathtt{X}^3 \times \sigma$. Then $\mathtt{e}$ specifies the 2-adic rational function $f(\sigma) = 8 \times \sigma$, and $\mathrm{rnf}(\mathtt{e}) = (\mathtt{X}^3 \times \sigma)/[\mathtt{1}]$. For the sake of readability, we will write expressions in their numeric interpretation. For example, the derivative

expression $(\mathtt{X}^2 + \mathtt{X}^3 \times \sigma)/[\mathtt{1}]$ will be denoted $4 + 8\sigma$. The diagram in Figure 3.3 shows the construction of our algorithm from the start specification 8σ. The

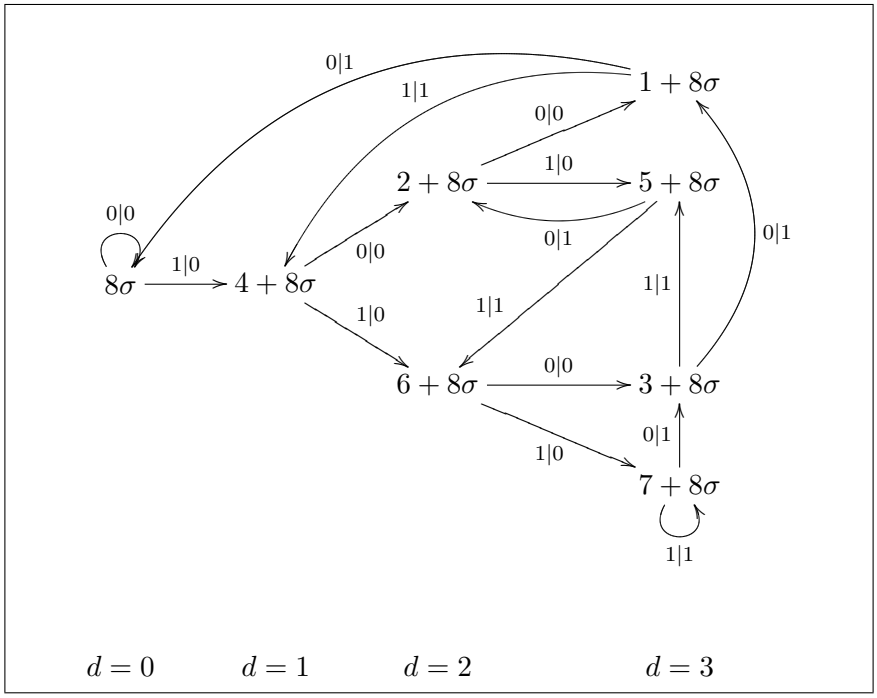


Figure 3.3: Mealy machine constructed from the 2-adic specification $\mathtt{e} = \mathtt{X}^3 \times \sigma$.

d-values below the diagram indicate the depth of the states in the constructed Mealy machine. At the start of iteration number d, the states with depth d will be contained in *New*. Hence in the example, after iteration 4 *New* will be empty, and the algorithm terminates. ◁

We have implemented our Mealy synthesis algorithm (cf. [54]) in the functional programming language Haskell [116]. Haskell is well suited for the kind of symbolic manipulation involved in the algorithm. In particular, the implementation was facilitated by Haskell's built-in pattern matching functionality and also its lazy evaluation mechanism, which allows coinductively defined datatypes.

3.5 Complexity

We now analyse the time complexity of the construction of Mealy machines given in Table 3.1 of Subsection 3.4.3. The main part of the analysis is common to specifications from both `Expr2Adic` and `ExprMod2`. We remark that all logarithms used in the analysis are of base 2.

The time complexity of the algorithm can be expressed in the following quantities:

M**:** the number of states in the constructed Mealy machine;

R**:** the time cost of computing and reducing derivatives to rational normal form;

E**:** the time cost of determining equivalence of two derivative expressions.

We also note that sets can be standardly implemented by lists. We will therefore include the cost of implementing the set operations as list operations.

3.5.1. Proposition. *The synthesis algorithm described in Table 3.1 runs in time* $O(RM + EM^3)$.

Proof. We have:

1. During the fixpoint computation, for every state q we compute and reduce the two derivatives of q exactly once (step 2). This yields a factor $M2R$.

2. The number of iterations is bounded by M, since at least one new state must be added in each iteration.

3. In each iteration we incur the following costs from the set operations:

   step 4: The lists *NewTrans* and *Prev* both have length at most $2M$. We can therefore carry out step 4 in time $O(EM^2)$.

   step 6: Remove duplicates (with respect to $\sim_{\mathcal{M}}$*) from New.* The length of the list *New* after step 4 is at most M. Removing duplicates from a list of length l takes at most l^2 number of comparisons. Hence step 6 can be carried out in time $O(EM^2)$.

Summing up, we obtain an overall complexity of $O(M2R + M(EM^2 + EM^2)) \in O(RM + EM^3)$. QED

3.5.2. Remark. *We would like to relate* M, R *and* E *to the input expression* $\mathtt{e}$*, but we observe the following.*

1. *For a constant, polynomial expression* $\mathtt{c}$*,* $\mathit{len}(\mathtt{c})$ *and* $\mathit{len}(\mathrm{cpnf}(\mathtt{c}))$ *are in general unrelated. For example, for* $\mathtt{n} \in \mathbb{N}$*, we have* $\mathit{len}(\mathtt{X^n} + -[\mathtt{1}]) = 4$*, but* $\mathit{len}(\mathrm{cpnf}(\mathtt{X^n} + -[\mathtt{1}])) = \mathit{len}([\mathtt{1}] + \mathtt{X} + \mathtt{X^2} + \ldots + \mathtt{X^{n-1}}) = 2n - 1$*. On the other hand,* $\mathit{len}([\mathtt{0}] + [\mathtt{0}] + [\mathtt{0}]) = 5$*, but* $\mathit{len}(\mathrm{cpnf}([\mathtt{0}] + [\mathtt{0}] + [\mathtt{0}])) = 1$*. This implies that for arbitrary* $\mathtt{e} \in \mathtt{Expr}$*, also* $\mathit{len}(\mathtt{e})$ *is unrelated with* $\mathit{len}(\mathrm{rnf}(\mathtt{e}))$.

2. *Given a specification* $\mathtt{e}$ *in rational normal form, the number of states in the Mealy machine constructed from* $\mathtt{e}$ *cannot be bounded in terms of* $len(\mathtt{e})$*. This is due to expressions of the form* $\mathtt{X^n}$ *having length 1. For example, for* $\mathtt{e} = \mathtt{X^n}/[\mathtt{1}]$*,* $len(\mathtt{e}) = 3$*, but the Mealy machine constructed from* $\mathtt{e}$ *will have* $n + 2$ *states.* ◁

Although Remark 3.5.2 tells us that we cannot in general relate M, R and E to the length of the input specification $\mathtt{e}$, we know from the results of sections 3.3.3 and 3.3.4 that we can bound M in terms of the numeric values and bitstream degree of the constant, polynomial subexpressions of the rational normal form of $\mathtt{e}$. These results can also be used to bound the length of the rational normal form of derivative expressions. In order to reach a bound on R and E it remains to show a bound on the length of derivative expressions before they are normalised, and determine the complexity of computing normal forms.

3.5.3. Lemma (Length of derivatives). *Let* $\mathtt{a} \in \mathtt{Bit}$*.*

1. *If* $\mathtt{c} \in \mathtt{CPolyExpr}$ *and* $\mathrm{cpnf}(\mathtt{c}) = \mathtt{c}$*, then:* $len(t(\mathtt{c})) \leq 2 \cdot len(\mathtt{c})$*.*
2. *If* $\mathtt{e} \in \mathtt{Expr}$ *is a rational function specification in rational normal form, then:* $len(t(\mathrm{inst}(\mathtt{e})(\mathtt{a}))) \in O(len(\mathtt{e}))$*.*
3. *If* $\mathtt{e} \in \mathtt{Expr}$ *is any expression then:* $len(t(\mathrm{inst}(\mathtt{e})(\mathtt{a}))) \in O(len(\mathtt{e})^2)$*.*

Proof. *Item 1:* For $\mathtt{c} \in \mathtt{CPoly2Adic}$, we first consider the case when $\mathrm{val}(\mathtt{c}) \geq 0$. Since $\mathtt{c}$ is assumed to be in normal form, i.e., $\mathrm{cpnf}(\mathtt{c}) = \mathtt{c}$, we know that $\mathtt{c}$ consists of k constant, atomic expressions and $k-1$ '+'-symbols, and $len(\mathtt{c}) = 2k-1$. Applying the derivative map t to $\mathtt{c}$ will produce 2 extra symbols for each '+', hence $len(t(\mathtt{c})) = 2(k-1)+2k-1 = 4k-3 \leq 2 \cdot len(\mathtt{c})$. In the case $\mathrm{val}(\mathtt{c}) < 0$, we have $\mathtt{c} = -\mathtt{d}$ with $\mathtt{d}$ in normal form, and $len(\mathtt{c}) = 2k$. Hence by the previous case and the definition of t, we get that $len(t(\mathtt{c})) = 3+len(t(\mathtt{d})) = 4k \leq 2 \cdot len(\mathtt{c})$. For $\mathtt{c} \in \mathtt{CPolyMod2}$, item 1 can be shown in a similar way.

Item 2: We only prove item 2 for the case $\mathtt{e} \in \mathtt{Expr2Adic}$ and $\mathtt{a} = \mathtt{1}$. The case for $\mathtt{a} = \mathtt{0}$ and $\mathtt{e} \in \mathtt{ExprMod2}$ can be shown using similar arguments. We have:

$$
\begin{aligned}
& len(t(\tfrac{\mathtt{e_d}+\mathtt{e_m}\times([\mathtt{1}]+\mathtt{X}\times\sigma)}{\mathtt{e_n}})) \\
= \;& len(t(\mathtt{e_d} + \mathtt{e_m} \times ([\mathtt{1}] + \mathtt{X} \times \sigma))) + len(t(\mathtt{e}_n)) + len(\mathtt{e}_n) + 4 \\
= \;& len(t(\mathtt{e}_d)) + len(t(\mathtt{e_m} \times ([\mathtt{1}] + \mathtt{X} \times \sigma))) + 2 + \\
& \quad len(t(\mathtt{e}_n)) + len(\mathtt{e}_n) + 4 \\
= \;& len(t(\mathtt{e}_d)) + (len(t(\mathtt{e}_m)) + len(([\mathtt{1}] + \mathtt{X} \times \sigma)) + \\
& \quad len(t(([\mathtt{1}] + \mathtt{X} \times \sigma))) + 4) + len(t(\mathtt{e}_n)) + len(\mathtt{e}_n) + 6
\end{aligned}
$$

$$\begin{aligned}
&= len(t(\mathtt{e}_d)) + len(t(\mathtt{e}_m)) + len(t(\mathtt{e}_n)) + len(\mathtt{e}_n) + 20 \\
\text{(item 1)} \quad &= 2len(\mathtt{e}_d) + 2len(\mathtt{e}_m) + 2len(\mathtt{e}_n) + len(\mathtt{e}_n) + 20 \\
&\leq 3(len(\mathtt{e}_d) + len(\mathtt{e}_m) + len(\mathtt{e}_n)) + 20 \\
&\leq 3len(\mathtt{e}) + 20 \\
&= O(len(\mathtt{e})).
\end{aligned}$$

Item 3: We merely provide a sketch. For arbitrary expressions $\mathtt{e}$ which may contain several occurrences of $\times$ and $/$, we have for all $\mathtt{a} \in \mathtt{Bit}$, that $len(\mathrm{inst}(\mathtt{e})(\mathtt{a})) \in O(len(\mathtt{e})^2)$, since each occurrence of $\times$ and $/$ will add $O(len(\mathtt{e}))$ symbols. The number of occurences of $\times$ and $/$ is bounded by $O(len(\mathtt{e}))$, hence the resulting derivative expression will have length $O(len(\mathtt{e})^2)$. QED

In order to determine the value R, we must determine the time complexity of reducing derivative expressions to normal form.

3.5.4. Proposition (Complexity of normalising derivatives).

1. *Let* $\mathtt{p} \in \mathtt{PolyExpr}$ *be a polynomial expression. Computing* $\mathrm{pnf}(\mathtt{p})$ *can be done in time* $2^{O(len(\mathtt{p}))}$.
2. *Let* $\mathtt{e} \in \mathtt{Expr}$ *be a rational function specification in rational normal form, and* $\mathtt{a} \in \mathtt{Bit}$. *Computing* $\mathrm{rnf}(t(\mathrm{inst}(\mathtt{e})(\mathtt{a})))$ *can be done in time* $2^{O(len(\mathtt{e}))}$.
3. *Let* $\mathtt{e} \in \mathtt{Expr}$ *be an arbitrary Mealy expression, and* $\mathtt{a} \in \mathtt{Bit}$. *Computing* $\mathrm{rnf}(t(\mathrm{inst}(\mathtt{e})(\mathtt{a})))$ *can be done in time* $2^{O(len(\mathtt{e})^2)}$.

Proof. *Item 1:* The time complexity of computing the polynomial normal form $\mathrm{pnf}(\mathtt{p})$ of a polynomial expression $\mathtt{p}$ is in the worst case exponential in $len(\mathtt{p})$ due to the duplication of subexpressions when applying the distributive law (e.g. $\mathtt{p} \times (\mathtt{q} + \mathtt{r}) = \mathtt{p} \times \mathtt{q} + \mathtt{p} \times \mathtt{r}$). All other manipulations in the computation of $\mathrm{pnf}(\mathtt{p})$ are polynomial in the length of the expression they are applied to. Hence the overall complexity will be $2^{O(len(\mathtt{e}))}$.

Item 2: When $\mathtt{e}$ is already in rational normal form, it follows from Definition 3.4.2 that $t(\mathrm{inst}(\mathtt{e})(\mathtt{a})) = \mathtt{p}/\mathtt{q}$ is a fraction with with $\mathtt{p}$ and $\mathtt{q}$ polynomial. Hence the rational normal form of $t(\mathrm{inst}(\mathtt{e})(\mathtt{a}))$ can be obtained by simply reducing $\mathtt{p}$ and $\mathtt{q}$ to polynomial normal form. From Lemma 3.5.3(2) we have that $len(t(\mathrm{inst}(\mathtt{e})(\mathtt{a}))) \in O(len(\mathtt{e}))$, Hence $len(\mathtt{p})$ and $len(\mathtt{q})$ are both of the order $O(len(\mathtt{e}))$. Consequently, due to item 1, computing the rational normal form of $t(\mathrm{inst}(\mathtt{e})(\mathtt{a}))$ can be done in time $2^{O(len(\mathtt{e}))}$.

Item 3: The case for arbitrary expressions is shown similarly to the previous item for rational function specifications, but now we apply Lemma 3.5.3(3), which yields a time complexity of $2^{O(len(\mathtt{e})^2)}$. QED

Deciding equivalence of two arbitrary expressions in rational normal form, $\mathtt{p}_1/\mathtt{q}_1$ and $\mathtt{p}_2/\mathtt{q}_2$ amounts to checking syntactic equality of $\mathrm{pnf}(\mathtt{p}_1 \times \mathtt{q}_2)$ and $\mathrm{pnf}(\mathtt{p}_2 \times \mathtt{q}_1)$. In order to analyse the time complexity of this decision method, we need to determine $\mathit{len}(\mathrm{pnf}(\mathtt{p} \times \mathtt{q}))$ given that $\mathtt{p}$ and $\mathtt{q}$ are already in normal form.

3.5.5. Lemma (Length of polynomial products).

1. *If* $\mathtt{c}, \mathtt{d} \in \mathtt{CPolyExpr}$ *are in normal form, then:* $\mathit{len}(\mathrm{cpnf}(\mathtt{c} \times \mathtt{d})) \leq \mathit{len}(\mathtt{c}) \cdot \mathit{len}(\mathtt{d})$.
2. *If* $\mathtt{p}, \mathtt{q} \in \mathtt{PolyExpr}$ *are in polynomial normal form, then:* $\mathit{len}(\mathrm{pnf}(\mathtt{p} \times \mathtt{q})) = O(\mathit{len}(\mathtt{p})^{3/2} \cdot \mathit{len}(\mathtt{q})^{3/2})$.

Proof. *Item 1:* Immediate from the shape of normal forms for constant, polynomial expressions. For example,

$$\begin{aligned} \mathit{len}(\mathrm{cpnf}((\mathtt{[1]} + \mathtt{X}) \times (\mathtt{[1]} + \mathtt{X}^2 + \mathtt{X}^4))) &= \mathit{len}(\mathtt{[1]} + \mathtt{X} + \mathtt{X}^2 + \mathtt{X}^3 + \mathtt{X}^4 + \mathtt{X}^5) \\ &= 11 \leq 3 \cdot 5 \\ &= \mathit{len}(\mathtt{[1]} + \mathtt{X}) \cdot \mathit{len}(\mathtt{[1]} + \mathtt{X}^2 + \mathtt{X}^4) \end{aligned}$$

Item 2: Suppose $\mathtt{p}$ and $\mathtt{q}$ have the following forms:

$$\mathtt{p} = \mathtt{c}_1 \times \sigma^{\mathtt{n}_1} + (\mathtt{c}_2 \times \sigma^{\mathtt{n}_2} + \ldots + \mathtt{c}_\mathtt{k} \times \sigma^{\mathtt{n}_\mathtt{k}})$$

$$\mathtt{q} = \mathtt{d}_1 \times \sigma^{\mathtt{m}_1} + (\mathtt{d}_2 \times \sigma^{\mathtt{m}_2} + \ldots + \mathtt{d}_\mathtt{l} \times \sigma^{\mathtt{m}_\mathtt{l}})$$

for $0 \leq \mathtt{n}_1 < \mathtt{n}_2 \ldots < \mathtt{n}_\mathtt{k}$ and $0 \leq \mathtt{m}_1 < \mathtt{m}_2 \ldots < \mathtt{m}_\mathtt{l}$. We note that in the above, an expression $\mathtt{c} \times \sigma^0$ should be read as $\mathtt{c}$.

We obtain $\mathrm{pnf}(\mathtt{p} \times \mathtt{q})$ by first applying distributivity. This yields a sum of kl terms of the form $\mathtt{c}_{\mathtt{n}_\mathtt{i}} \times \mathtt{d}_{\mathtt{m}_\mathtt{j}} \times \sigma^{\mathtt{n}_\mathtt{i}+\mathtt{m}_\mathtt{j}}$. The polynomial normal form of $\mathtt{p} \times \mathtt{q}$ is now obtained from this distributive form by grouping together terms which have the same σ-exponent, and reducing the resulting sum of coefficients to normal form. However, $\mathrm{pnf}(\mathtt{p} \times \mathtt{q})$ could also be computed by first normalising all coefficients, (use notation: $\mathtt{f}_{\mathtt{i},\mathtt{j}} = \mathrm{cpnf}(\mathtt{c}_{\mathtt{n}_\mathtt{i}} \times \mathtt{d}_{\mathtt{m}_\mathtt{j}})$) and repeating the following process: for any two terms $\mathtt{f}_{\mathtt{i},\mathtt{j}} \times \sigma^{\mathtt{n}_\mathtt{i}+\mathtt{m}_\mathtt{j}}$ and $\mathtt{f}_{\mathtt{i}',\mathtt{j}'} \times \sigma^{\mathtt{n}_{\mathtt{i}'}+\mathtt{m}_{\mathtt{j}'}}$ where $\mathtt{n}_\mathtt{i} + \mathtt{m}_\mathtt{j} = \mathtt{n}_{\mathtt{i}'} + \mathtt{m}_{\mathtt{j}'}$, replace them with the term $\mathrm{cpnf}(\mathtt{f}_{\mathtt{i},\mathtt{j}} + \mathtt{f}_{\mathtt{i}',\mathtt{j}'}) \times \sigma^{\mathtt{n}_\mathtt{i}+\mathtt{m}_\mathtt{j}}$. There are finitely many terms, so this process will surely terminate. Moreover, we observe that such a replacement decreases the length of the total expression, since:

$$\begin{aligned} & \mathit{len}(\mathrm{cpnf}(\mathtt{f}_{\mathtt{i},\mathtt{j}} + \mathtt{f}_{\mathtt{i}',\mathtt{j}'}) \times \sigma^{\mathtt{n}_\mathtt{i}+\mathtt{m}_\mathtt{j}}) \\ \leq\ & \mathit{len}(\mathrm{cpnf}(\mathtt{f}_{\mathtt{i},\mathtt{j}} + \mathtt{f}_{\mathtt{i}',\mathtt{j}'})) + (n_i + m_j) + 1 \\ \leq\ & \mathit{len}(\mathtt{f}_{\mathtt{i},\mathtt{j}}) + \mathit{len}(\mathtt{f}_{\mathtt{i}',\mathtt{j}'}) + (n_i + m_j) + 1 \\ \leq\ & \mathit{len}(\mathtt{f}_{\mathtt{i},\mathtt{j}}) + \mathit{len}(\mathtt{f}_{\mathtt{i}',\mathtt{j}'}) + 2 \cdot (n_i + m_j) + 4 \\ =\ & \mathit{len}(\mathtt{f}_{\mathtt{i},\mathtt{j}} \times \sigma^{\mathtt{n}_\mathtt{i}+\mathtt{m}_\mathtt{j}} + \mathtt{f}_{\mathtt{i}',\mathtt{j}'} \times \sigma^{\mathtt{n}_{\mathtt{i}'}+\mathtt{m}_{\mathtt{j}'}}) \end{aligned}$$

This implies that the length of $\mathrm{pnf}(\mathtt{p} \times \mathtt{q})$ is at most the length of the distributive form with normalised coefficients, i.e.

$$\begin{array}{lll} len(\mathrm{pnf}(\mathtt{p} \times \mathtt{q})) & \leq & \Sigma_{i=1}^{k}\Sigma_{j=1}^{l} len(\mathtt{f_{i,j}} \times \sigma^{\mathtt{n_i}+\mathtt{m_j}}) \\ & \leq_{(\text{item 1})} & \Sigma_{i=1}^{k}\Sigma_{j=1}^{l} len(\mathtt{c_i}) \cdot len(\mathtt{d_j}) + n_i + m_j + 1. \end{array}$$

Let $l_p = len(\mathtt{p})$ and $l_q = len(\mathtt{q})$. We then have $len(\mathtt{c_i}), n_i \leq l_p$ and $len(\mathtt{d_j})$, $m_j \leq l_q$. As $\mathtt{p}$ is in polynomial normal form, l_p is greater than or equal to the number of occurrences of σ plus the number of occurrences of $+$, that is,

$$\begin{array}{rcl} l_p & \geq & \Sigma_{i=1}^{k} n_i + (k-1) \\ & \geq & \Sigma_{i=1}^{k} i \; + \; (k-1) \\ & = & \frac{1}{2}(k^2+k) + (k-1) = \frac{1}{2}k^2 + \frac{3}{2}k - 1 \\ & \geq & \frac{1}{2}k^2 \text{ for all } k \geq 1. \end{array}$$

Hence $k \leq \sqrt{2l_p}$. Similarly, we find that $l \leq \sqrt{2l_q}$. We now conclude that

$$\begin{array}{rcl} len(\mathrm{pnf}(\mathtt{p} \times \mathtt{q})) & \leq & kl \cdot (l_p \cdot l_q + l_p + l_q + 1) \\ & \leq & \sqrt{2l_p} \cdot \sqrt{2l_q} \cdot (l_p \cdot l_q + l_p + l_q + 1) \\ & = & O({l_p}^{3/2} \cdot {l_q}^{3/2}). \end{array}$$

QED

3.5.6. Proposition (Complexity of deciding equivalence).

1. *Let* $\mathtt{e_1}, \mathtt{e_2} \in \mathtt{Expr}$ *be in rational normal form (and of the same type). Deciding* $\mathtt{e_1} \sim_{\mathcal{M}} \mathtt{e_2}$ *can be done in time* $2^{O(len(\mathtt{e_1})+len(\mathtt{e_2}))}$.

2. *Let* $\mathtt{e}, \mathtt{e_1}, \mathtt{e_2} \in \mathtt{Expr}$ *be in rational normal form (and of the same type), and assume that* $\mathtt{e}$ *is a rational function specification, and* $\mathtt{e_1}, \mathtt{e_2}$ *are derivative expressions of* $\mathtt{e}$. *Deciding* $\mathtt{e_1} \sim_{\mathcal{M}} \mathtt{e_2}$ *can be done in time* $O(len(\mathtt{e_1}) + len(\mathtt{e_2}))$.

Proof. *Item 1:* Assume expressions $\mathtt{e_1} = \mathtt{p_1}/\mathtt{q_1}$ and $\mathtt{e_2} = \mathtt{p_2}/\mathtt{q_2}$ are in rational normal form. Both $\mathtt{p_1} \times \mathtt{q_2}$ and $\mathtt{p_2} \times \mathtt{q_1}$ have length $O(len(\mathtt{e_1}) + len(\mathtt{e_2}))$, hence by Proposition 3.5.4(1), computing $\mathrm{pnf}(\mathtt{p_1} \times \mathtt{q_2})$ and $\mathrm{pnf}(\mathtt{p_2} \times \mathtt{q_1})$ can be done in time $2^{O(len(\mathtt{e_1})+len(\mathtt{e_2}))}$. From Lemma 3.5.5(2) we have that $len(\mathrm{pnf}(\mathtt{p_1} \times \mathtt{q_2})) \in O(len(\mathtt{p_1})^{3/2} \cdot len(\mathtt{q_2})^{3/2}) \in O(len(\mathtt{e_1})^{3/2} \cdot len(\mathtt{e_2})^{3/2})$. Similarly, we find that $len(\mathrm{pnf}(\mathtt{p_2} \times \mathtt{q_1})) \in O(len(\mathtt{e_1})^{3/2} \cdot len(\mathtt{e_2})^{3/2})$. Checking syntactic equality of expressions is linear in the length of the expressions, hence checking equivalence of $\mathtt{e_1}$ and $\mathtt{e_2}$ has time complexity: $2^{O(len(\mathtt{e_1})+len(\mathtt{e_2}))} + O(len(\mathtt{e_1})^{3/2} \cdot len(\mathtt{e_2})^{3/2}) \in 2^{O(len(\mathtt{e_1})+len(\mathtt{e_2}))}$.

Item 2: If $\mathtt{e}$ is a rational function specification, then $\mathrm{rnf}(\mathtt{e}) = \mathtt{p}/\mathtt{q}$ where $\mathtt{q}$ is a constant, polynomial expression. Hence by the definition of the Mealy

structure on $\mathtt{Expr}$ (cf. Definition 3.4.2 and Definition 3.4.3), the rational normal form of any two derivative expressions $\mathtt{e}_1$ and $\mathtt{e}_2$ have the shape $\mathrm{rnf}(\mathtt{e}_1) = \mathtt{p}_1/\mathtt{q}$ and $\mathrm{rnf}(\mathtt{e}_2) = \mathtt{p}_2/\mathtt{q}$, i.e., they all have the same constant denominator $\mathtt{q}$. To determine equivalence of such rational normal form expressions, there is no need to compute and compare $\mathrm{pnf}(\mathtt{p}_1 \times \mathtt{q})$ and $\mathrm{pnf}(\mathtt{p}_2 \times \mathtt{q})$, since $\mathtt{p}_1/\mathtt{q} \sim_{\mathcal{M}} \mathtt{p}_2/\mathtt{q}$ if and only if $\mathtt{p}_1 = \mathtt{p}_2$. This latter syntactic comparison can be done in time $O(\mathit{len}(\mathtt{p}_1) + \mathit{len}(\mathtt{p}_2)) \in O(\mathit{len}(\mathtt{e}_1) + \mathit{len}(\mathtt{e}_2))$. QED

3.5.7. Remark. For arbitrary $\mathtt{e}_1 = \mathtt{p}_1/\mathtt{q}_1$ and $\mathtt{e}_2 = \mathtt{p}_2/\mathtt{q}_2$ in rational normal form, checking equality of numerators is not sound. In our program, we have therefore chosen to check both $\mathtt{p}_1 = \mathtt{p}_2$ and $\mathtt{q}_1 = \mathtt{q}_2$, which clearly still has linear time complexity. When deciding equivalence of derivatives of rational function specifications, this check is both sound and complete. Our program uses by default the exponential time check which is sound and complete for all expressions. The optimised, linear equivalence check must be enabled by the user (see end of subsection 3.4.3). ◁

We are now almost ready to state and prove the time complexity of our synthesis algorithm for rational 2-adic and mod-2 specifications. The length of expressions is useful for the analysis of the symbolic computations carried out by our program, but the complexity of an algorithm is usually formulated in terms of the amount of memory (e.g. bits) needed to represent the input $\mathtt{e}$. We denote this size measure by $\mathrm{spc}(\mathtt{e})$.

3.5.8. Definition. Let $\mathbf{k} \in \mathbb{N}$ be such that we can encode the arithmetic operation symbols and the expressions $[\mathtt{0}], [\mathtt{1}], \mathtt{X}, \sigma$ with $\mathbf{k}$ bits. Hence $\mathbf{k} \geq 2$. We define $\mathrm{spc}\colon \mathtt{Expr} \to \mathbb{N}$ as follows.

$$\begin{array}{lcll}
\mathrm{spc}(\mathtt{c}) & = & \mathbf{k} & \text{if } \mathtt{c} \in \{[\mathtt{0}], [\mathtt{1}], \sigma\}, \\
\mathrm{spc}(X^n) & = & \mathbf{k} + \log(n), & \\
\mathrm{spc}(\mathtt{neg}\,\mathtt{e}) & = & \mathbf{k} + \mathrm{spc}(\mathtt{e}) & \text{if } \mathtt{neg} \in \{-, \ominus\}, \\
\mathrm{spc}(\mathtt{e}_1\,\mathtt{op}\,\mathtt{e}_2) & = & \mathbf{k} + \mathrm{spc}(\mathtt{e}_1) + \mathrm{spc}(\mathtt{e}_2) & \text{if } \mathtt{op} \in \{+, \times, /, \oplus, \otimes, \oslash\}.
\end{array}$$ ◁

We will use the results on the size of minimal realisations from subsections 3.3.3 and 3.3.4. The upper bound on the automaton size for rational 2-adic functions (Theorem 3.3.10) is formulated in terms of the numeric interpretation of 2-adic functions, and the analogous result for rational mod-2 functions (Theorem 3.3.21) is formulated in terms of bitstream degree. In the next lemma we relate these quantities to *len* and spc of constant, polynomial expression.

First we introduce some notation. Given an integer $z \in \mathbb{Z}$, we write $\mathtt{e}_z$ for the unique constant, polynomial 2-adic expression in normal form which satisfies $\mathrm{val}(\mathtt{e}_z) = z$ and $\mathrm{cpnf}(\mathtt{e}_z) = \mathtt{e}_z$. Moreover, for any constant, polynomial

expression $\mathtt{c}$ we write $\mathrm{Beh}(\mathtt{c})$ not only for the constant bitstream function $f_{\mathtt{c}}$ specified by $\mathtt{c}$, but also for the bitstream behaviour of $\mathtt{c}$.

We now show how $\mathit{len}(\mathtt{c})$ and $\mathrm{spc}(\mathtt{c})$ for a constant, polynomial expression $\mathtt{c}$ are related to the numeric value or bitstream degree of $\mathrm{Beh}(\mathtt{c})$.

3.5.9. Lemma.

1. *For all* $\mathtt{c} \in \mathtt{CPoly2Adic}$ *in normal form:*
   (a) $\mathit{len}(\mathtt{c}) \leq 2 \cdot \log(|\operatorname{val}(\mathtt{c})|) + 2$, *if* $\operatorname{val}(\mathtt{c}) \neq 0$, *and*
   (b) $|\operatorname{val}(\mathtt{c})| \leq 2^{2^{\mathrm{spc}(\mathtt{c})}}$.
2. *For all* $\mathtt{c} \in \mathtt{CPolyMod2}$ *in normal form:*
   (a) $\mathit{len}(\mathtt{c}) \leq 2 \cdot \#(\mathrm{Beh}(\mathtt{c})) - 1$, *and*
   (b) $\#(\mathrm{Beh}(\mathtt{c})) \leq 2^{\mathrm{spc}(\mathtt{c})}$, *if* $\#(\mathrm{Beh}(\mathtt{c})) \geq 2$.

Proof. *Item (1.a):* An expression $\mathtt{c} \in \mathtt{CPoly2Adic}$ in normal form which maximises $\mathit{len}(\mathtt{c})$ with respect to $|\operatorname{val}(\mathtt{c})|$ has the form $\mathtt{c} = -([\mathtt{1}] + \mathtt{X} + \mathtt{X}^{\mathtt{2}} + \ldots + \mathtt{X}^{\mathtt{n}})$ for some $\mathtt{n} \in \mathbb{N}$. For such a $\mathtt{c}$, we have: $|\operatorname{val}(\mathtt{c})| = 2^{n+1} - 1$ and $\mathit{len}(\mathtt{c}) = 2(n+1) \leq 2 \cdot (\log(|\operatorname{val}(\mathtt{c})|) + 1)$.

Item (1.b): On the other hand, a constant, polynomial $\mathtt{c} \in \mathtt{Expr2Adic}$ in normal form for which $|\operatorname{val}(\mathtt{c})|$ is maximal with respect to $\mathrm{spc}(\mathtt{c})$ is of the form $\mathtt{X}^{\mathtt{n}}$. In this case, $\log(n) \leq \mathrm{spc}(\mathtt{c})$ which implies $\log(|\operatorname{val}(\mathtt{c})|) = n \leq 2^{\mathrm{spc}(\mathtt{c})}$, and hence $|\operatorname{val}(\mathtt{c})| \leq 2^{2^{\mathrm{spc}(\mathtt{c})}}$.

Item (2.a): A constant, polynomial mod-2 expression $\mathtt{c}$ in normal form which maximises $\mathit{len}(\mathtt{c})$ with respect to $\#(\mathrm{Beh}(\mathtt{c}))$ has the form $\mathtt{c} = [\mathtt{1}] \oplus \mathtt{X} \oplus \ldots \oplus \mathtt{X}^{\mathtt{n}}$ for $\mathtt{n} = \#(\mathrm{Beh}(\mathtt{c})) - 2$, and we find that: $\mathit{len}(\mathtt{c}) = \mathit{len}([1] \oplus X \oplus \ldots \oplus X^n) = 2n + 1 = 2(n+1) - 3 = 2 \cdot \#(\mathrm{Beh}(\mathtt{c})) - 3$.

Item (2.b): Similarly, $\#(\mathrm{Beh}(\mathtt{c}))$ is maximal with respect to $\mathrm{spc}(\mathtt{c})$ if $\mathtt{c} \in \mathtt{Const}$. If $\mathtt{c}$ is [0] or [1], then $\#(\mathrm{Beh}(\mathtt{c})) \leq 2 \leq 2^k$. If $\mathtt{c} = \mathtt{X}^{\mathtt{n}}$ for $n \geq 1$, then

$$\#(\mathrm{Beh}(\mathtt{X}^{\mathtt{n}})) = n + 2 \leq 2^{\mathtt{k}} \cdot n = 2^{\mathtt{k}+\log(n)} = 2^{\mathrm{spc}(\mathtt{X}^{\mathtt{n}})}. \qquad \text{QED}$$

We can now state and prove the time complexity of the Mealy machine construction from rational 2-adic specifications.

3.5.10. Theorem. *Let* $\mathtt{e} = \frac{\mathtt{e}_d + \mathtt{e}_m \times \sigma}{\mathtt{e}_n}$ *be a rational 2-adic function specification in rational normal form. The synthesis algorithm from Table 3.1 constructs a minimal realisation of* $\mathrm{Beh}(\mathtt{e})$ *in time* $O(K^c)$ *for* $K = |d| + |m| + |n|$ *and some* $c \geq 4$, *or in terms of input size, in time* $2^{O(2^{\mathrm{spc}(\mathtt{e})})}$.

PROOF. From Theorem 3.3.10, we have that: $M \leq |d| + |m| + |n| = K$. Using Lemma 3.5.9(1a), we find that

$$\begin{aligned} len(\mathtt{e}) &= len(\mathtt{e}_d) + len(\mathtt{e}_m) + len(\mathtt{e}_n) + 4 \\ &\leq 2 \cdot (\log(|d|) + \log(|m|) + \log(|n|)) + 10 \\ &\leq 6 \cdot \log(K) + 10. \end{aligned}$$

From Lemma 3.3.6 and Definition 3.4.3 it follows that any derivative of $\mathtt{e}$ has rational normal form $\mathtt{e}_\mathtt{w} = \frac{\mathtt{e}_{d_w} + \mathtt{e}_m \times \sigma}{\mathtt{e}_n}$, where $|d_w| \leq K$. Hence by Lemma 3.5.9(1a),

$$\begin{aligned} len(\mathtt{e}_\mathtt{w}) &= len(\mathtt{e}_{d_w}) + len(\mathtt{e}_m) + len(\mathtt{e}_n) + 4 \\ &\leq 2 \cdot (\log(K) + \log(|m|) + \log(|n|)) + 10 \\ &\leq 6 \cdot \log(K) + 10. \end{aligned}$$

From Proposition 3.5.4(2) it now follows that we can compute and reduce to rational normal form any derivative of $\mathtt{e}$ in time $2^{O(\log(K))}$. That is, $R \in 2^{O(\log(K))}$, which means there exists a $c_R \geq 1$ such that $R \leq 2^{c_R \log(K)}$ for sufficiently large K. Similarly, from Proposition 3.5.6(2), we find that comparing normalised derivatives costs $E \in O(\log(K))$, i.e., there exists a $c_E \geq 1$ such that $E \leq c_E \cdot \log(K)$ for sufficiently large K. Hence by Proposition 3.5.1, the overall complexity of the construction for $\mathtt{e}$ is

$$\begin{aligned} O(MR + EM^3) &= O(K \cdot 2^{c_R \log(K)} + c_E \cdot \log(K) \cdot K^3) \\ &= O(K^{1+c_R} + c_E \cdot \log(K) \cdot K^3) \\ &\in O(K^c) \quad \text{for some } c \geq 4. \end{aligned}$$

Finally, it follows from Lemma 3.5.9(1b) that

$$K = |d| + |m| + |n| \;\leq\; 2^{2^{\mathrm{spc}(\mathtt{e}_d)}} + 2^{2^{\mathrm{spc}(\mathtt{e}_m)}} + 2^{2^{\mathrm{spc}(\mathtt{e}_n)}} \;\leq\; 3 \cdot 2^{2^{\mathrm{spc}(\mathtt{e})}},$$

and hence $K^c \in 2^{O(2^{\mathrm{spc}(\mathtt{e})})}$ for any $c \geq 4$. QED

Finally, we give the time complexity for our Mealy synthesis algorithm for rational mod-2 specifications.

3.5.11. THEOREM. *Let* $\mathtt{e} = \frac{\mathtt{d} \oplus \mathtt{q} \otimes \sigma}{\mathtt{r}}$ *be a rational mod-2 function specification in rational normal form. The synthesis algorithm from Table 3.1 constructs a minimal realisation of* $\mathrm{Beh}(\mathtt{e})$ *in time* $2^{O(K)}$ *where*

$$K = \max\{\#(\mathrm{Beh}(\mathtt{d})), \#(\mathrm{Beh}(\mathtt{q})), \#(\mathrm{Beh}(\mathtt{r}))\},$$

or alternatively, in terms of input size, in time $2^{O(2^{\mathrm{spc}(\mathtt{e})})}$.

PROOF. From Theorem 3.3.21, we have that: $M \leq 1 + 2^{K-1}$. We now have, using Lemma 3.5.9(2a):

$$\begin{aligned} len(\mathtt{e}) &= len(\mathtt{d}) + len(\mathtt{q}) + len(\mathtt{r}) + 4 \\ &\leq 2 \cdot (\#(\mathrm{Beh}(\mathtt{d})) + \#(\mathrm{Beh}(\mathtt{q})) + \#(\mathrm{Beh}(\mathtt{r}))) - 5 \\ &\leq 6K - 5 \end{aligned}$$

For $K > 1$, it follows from Lemma 3.3.19 that the rational normal form of any derivative of $\mathtt{e}$ is of the form $\mathtt{e}_\mathtt{w} = \frac{\mathtt{d}_\mathtt{w} + \mathtt{q} \times \sigma}{\mathtt{r}}$, where $\#(\mathrm{Beh}(\mathtt{d}_\mathtt{w})) \leq K-1$. Hence by Lemma 3.5.9(2a), $len(\mathtt{d}_\mathtt{w}) \leq 2(K-1) - 3 = 2K - 5$, and we have for all $w \in 2^+$:

$$\begin{aligned} len(\mathtt{e}_\mathtt{w}) &= len(\mathtt{d}_\mathtt{w}) + len(\mathtt{q}) + len(\mathtt{r}) + 4 \\ &\leq (2K-5) + (2K-3) + (2K-3) + 4 \\ &= 6K - 7. \end{aligned}$$

From Proposition 3.5.4(2) it now follows that we can compute and reduce to rational normal form any derivative of $\mathtt{e}$ in time $2^{O(K)}$. That is, $R \in 2^{O(K)}$, which means there exists a $c_R \geq 1$ such that $R \leq 2^{c_R \cdot K}$ for sufficiently large K. Similarly, from Proposition 3.5.6(2), we find that comparing normalised derivatives costs $E \in O(K)$, i.e., there exists a $c_E \geq 1$ such that $E \leq c_E \cdot K$ for sufficiently large K. Hence by Proposition 3.5.1, the overall complexity of the construction for $\mathtt{e}$ is

$$\begin{aligned} O(MR + EM^3) &= O((1 + 2^K) \cdot 2^{c_R \cdot K} + c_E \cdot K \cdot (1 + 2^K)^3) \\ &\in O(2^{(1+c_R) \cdot K} + 2^{3K}) \\ &\in 2^{O(K)}. \end{aligned}$$

To prove the doubly-exponential bound in $\mathrm{spc}(\mathtt{e})$, it suffices to show that $K \leq 2^{\mathrm{spc}(\mathtt{e})}$. We have from Lemma 3.5.9(2b) that $\#(\mathrm{Beh}(\mathtt{d})) \leq 2^{\mathrm{spc}(\mathtt{d})} \leq 2^{\mathrm{spc}(\mathtt{e})}$. Similarly, for $\mathtt{q}$ and $\mathtt{r}$. It follows that, $K \leq 2^{\mathrm{spc}(\mathtt{e})}$, QED

3.5.12. REMARK. The upper bounds of Theorems 3.5.10 and 3.5.11 are doubly-exponential in $\mathrm{spc}(\mathtt{e})$ due to our $O(\log(n))$-size representation of $\mathtt{X}^\mathtt{n}$. If we had chosen not to include powers of X as atomic expressions, we would have a length function slen for which $\mathrm{slen}(\mathtt{X}^\mathtt{n}) = 2n - 1$, and we have:

1. $|\,\mathrm{val}(\mathtt{c})| \leq 2^{\mathrm{slen}(\mathtt{c})}$ for all $\mathtt{c} \in \mathtt{CPoly2Adic}$.
2. $\#(\mathrm{Beh}(\mathtt{c})) \leq \mathrm{slen}(\mathtt{c})$ for all $\mathtt{c} \in \mathtt{CPolyMod2}$.

This would lead to a time complexity of $2^{O(\mathrm{slen}(\mathtt{e}))}$ for rational 2-adic and mod-2 specifications $\mathtt{e}$ in normal form, but only under the assumption that the length of derivatives and computing polynomial normal forms would be of the same complexity as with our $O(\log(n))$-size representation of $\mathtt{X}^\mathtt{n}$. The amount of memory space needed to represent an expression $\mathtt{e}$ would now be $O(\mathrm{slen}(\mathtt{e}))$. ◁

3.6 Conclusion

Building on [132, 55], we have shown how to construct a minimal Mealy realisation from a rational specification in 2-adic or mod-2 bitstream arithmetic. The construction is an example of what we call coalgebraic synthesis, since it is based on the idea of defining a coalgebraic structure on the set of specifications, and constructing realisations by computing subcoalgebras. The correctness of our Mealy machine construction is clear from the coalgebraic modelling of Mealy machines and the coinductive definition of the bitstream operations. Based on experiments using our implementation of the synthesis algorithm, we were able to conjecture and prove upper bounds on the size of the constructed automaton in terms of the parameters of the specification.

The relationship between 2-adic numbers and digital circuits has been investigated by Vuillemin [157, 156, 155]. In [155], Vuillemin shows how to construct from a 2-adic function specification a synchronous decision diagram, a type of structure which relates to causal bitstream functions and sequential circuits, roughly as binary decision diagrams relate to Boolean functions and combinational circuits. This construction is also based on the notion of derivative, but the resulting structure is closer to an actual circuit than a Mealy machine. Vuillemin [155] does not report on the complexity of the construction.

Closely related to the work presented in this chapter, is the coalgebraic synthesis of Mealy machines from logic specifications by Bonsangue, Rutten & Silva [27]. Here, Mealy machines are specified in a modal language extended with a single fixpoint operator. This language is expressive, meaning that every finite Mealy machine can be specified by a formula, and conversely, every (consistent) formula is realisable. However, the constructed Mealy machine is not necessarily minimal. The results from [27] have been generalised to coalgebras for Kripke polynomial functors in [28]. We point out that in [27, 28] the specification language is derived from the functor, and its expressions are therefore in close correspondence with the semantic structure. The bitstream languages of this chapter are more high-level, and most likely not useful for specifying non-arithmetic circuit behaviours such as n-bit registers and queues, although we do not exclude that it is possible to do so.

Synthesis of Mealy (or Moore) type automata from logic specifications has a long history [38, 31, 117, 149, 64, 79]. The main idea here is that a logic formula φ (in e.g. monadic second order logic or linear time temporal logic) specifies a relation R_φ between input and output streams, and from φ one can construct an automaton $\mathcal{A}_\varphi$ which accepts Mealy behaviours f which satisfy φ, meaning that for all input streams σ, $R_\varphi(\sigma, f(\sigma))$ holds. The actual construction is carried out as a constructive nonemptiness test of $\mathcal{A}_\varphi$. This automata-theoretic approach to synthesis differs from coalgebraic synthesis in the following ways: (i) A monadic second order formula φ defines a relational requirement which may have several

Mealy behaviours as a solution, whereas a specification e in coalgebraic synthesis corresponds to at most one Mealy behaviour, namely $\mathrm{Beh}(\mathsf{e})$. (ii) The automaton $\mathcal{A}_\varphi$ described above has the property that the finite-state requirement is built in: If $\mathcal{A}_\varphi$ accepts some Mealy coalgebra, then it accepts one with finitely many states, and such a solution is constructed during the nonemptiness test. In coalgebraic synthesis, in general, and for our arithmetic bitstream expressions in particular, we need to know that a specification has a finite realisation before we start our construction, since otherwise the construction will not terminate. (iii) The automaton constructions and transformations carried out during automata-theoretic synthesis are of a considerable (conceptual and computational) complexity, whereas the coalgebraic construction of Mealy machines by computing derivatives is direct and conceptually simple. In coalgebraic Mealy synthesis, the complexity arises from the need to decide equivalence of expressions.

An interesting combination of synthesis using derivatives and automata-based techniques is given by Redziejowski in [121] where deterministic ω-automata are constructed using derivatives of ω-regular expressions. This construction, however, is not a straightforward generalisation of Brzozowski's algorithm, and it involves techniques similar to those used in the determinisation of Büchi automata [134]. It would be interesting to find out how Redziejowski's construction relates to coalgebraic synthesis. Coalgebraic synthesis of ω-automata presupposes a coalgebraic modelling of ω-automata. Even for the class of deterministic Büchi automata such a coalgebraic modelling seems not to exist, at least not over the category of sets and functions. We base this claim on the observation that there exist equivalent deterministic Büchi automata with a minimal number of states, but without any intuitive notion of bisimulation linking them. Finally we mention that Mealy synthesis is also carried out in the settings of learning [105] and neural networks [144].

In general, the most challenging part of coalgebraic synthesis seems to be the need for efficient equational reasoning over the chosen specification language. In fact, the ACI-equivalence check used in Brzozowski's method is by some considered too inefficient to be practical. Various techniques have been developed to avoid this problem in the construction of DFA's from regular expressions. We give a brief overview, since the ideas behind these techniques could perhaps be generalised and help make coalgebraic synthesis more efficient. Berry & Sethi [22] combine the elegance of Brzozowski's method with the efficiency of an earlier construction by McNaughton & Yamada [95]. Berry & Sethi prove as their main theorem that if e is a regular expression in which all letters are unique, then all derivatives of e are unique up to ACI-equivalence. Hence Brzozowski's algorithm can be carried out on e without the need for the expensive ACI-equivalence check. In their improved algorithm, starting from a regular expression e, all letter occurrences in e are subscripted to yield an e' in which

all letters are unique. Then the fast version of Brzozowski's construction is carried out on e'. Removing the letter subscripts from the resulting automaton $\mathbb{A}'$, yields a nondeterministic automaton $\mathbb{A}$ which accepts the language denoted by e. The final step is to determinise $\mathbb{A}$ using the subset construction. Antimirov [9] improves on the efficiency of Brzozowski's method by introducing partial derivatives of regular expressions which give rise to a direct construction of nondeterministic finite automata from regular expressions. Subsequently, Champarnaud & Ziadi [33] have combined Antimirov's construction with the method of Berry & Sethi [22] by defining canonical derivatives of regular expressions, leading to yet another improvement in complexity.

Another way of characterising minimal realisations of regular languages, which seems closely related to coalgebraic synthesis, is found in the work by Gehrke et al. [45]. The authors consider the Boolean algebra $\mathsf{Reg}(A)$ of regular languages over A extended with the operations of left and right residuals of languages, $\backslash$ and $/$, and show that it is dual to the topological space of profinite words over A extended with product. Given a regular language $L \in \mathsf{Reg}(A)$, the subalgebra generated by $\{L\}$ in $(\mathsf{Reg}(A^*), \backslash, /)$ has as its dual space the syntactic ordered monoid of L, and hence it corresponds directly to a minimal deterministic automaton accepting L. We would like to understand better the connection between the work in [45] and the coalgebraic view on regular languages (cf. Subsection 2.4.2).

We conclude by mentioning that bialgebras could play an important role in coalgebraic synthesis. Informally stated, a bialgebra consists of an algebra and a coalgebra on the same carrier set which satisfy certain compatibility requirements. The combination of algebraic and coalgebraic structure on the set of specifications is central to coalgebraic synthesis. It would be good to understand the exact role played by bialgebras in coalgebraic synthesis, and whether results on bialgebras are useful for proving properties of coalgebraic synthesis. For regular expressions and deterministic automata these questions have been addressed by Jacobs [70].

Chapter 4

Coalgebraising subsequential transducers

4.1 Introduction

Subsequential transducers generalise classic deterministic automata as well as Mealy [96] and Moore [101] type state machines. They can be described as input deterministic automata which produce output words on transitions and terminal output at accepting states. The input/output behaviour of a subsequential transducer is the partial word function it computes, and whose domain consists of the words accepted by the underlying automaton. This combination of language recognition and transduction makes subsequential transducers useful in areas such as lexical analysis, coding theory, and more recently, in speech and language processing (cf. [98, 99]).

Schützenberger [139] introduced subsequential transducers as a generalisation of sequential transducers by adding the features of non-accepting states and terminal output. At this time, the theory of sequential transducers and their behaviours was already well developed as witnessed by Eilenberg [41]. Subsequential transducers have been studied, in particular, by Choffrut [36, 37] who showed that subsequential transducers can be minimised via a normalisation construction; that any partial word function is computed by a minimal subsequential transducer (possibly with infinitely many states); and that the Ginsburg-Rose characterisation theorem for sequential functions can be generalised to the subsequential case. Other results on subsequential transducers may be found in [17, 18, 29].

In the abovementioned work, subsequential transducers have been studied from the algebraic perspective on automata. The aim of this chapter is to place subsequential transducers in a coalgebraic framework, and to gain a deeper understanding of existing notions and constructions on subsequential transducers by investigating whether they are instances of more general mathematical (or coalgebraic) concepts. Deterministic automata and Mealy/Moore machines are by now well-known examples of automata which can be modelled as coalgebras, (see the Introduction and Chapter 3). The starting point of our investigation

is whether this coalgebraic modelling can be generalised to the class of subsequential transducers.

This chapter has three main parts. In the first part (Section 4.3), we give the formal definition of subsequential transducers, subsequential structures, their morphisms and behaviour. Next, we investigate the relationship between various subcategories of subsequential transducers and structures. We will see that the classes of coaccessible, normalised and minimal subsequential structures form a sequence of nested, full, reflective subcategories of the category of all subsequential structures. Each move to a reflective subcategory can be seen as a step towards an optimal representation. These results give a clear break-down of the essential steps of minimisation and parallel known results for deterministic automata (cf. [3, 5]).

In the second part (Section 4.4), we turn to the coalgebraic modelling. We will see that, although subsequential structures as objects have the type of coalgebras for a functor S, they cannot properly be regarded as coalgebras, in general, since the associated notion of S-coalgebra morphism is too strict with respect to the intended word function semantics. However, we will show that in the subcategory $\mathsf{NSubseq}$ of normalised subsequential structures, subsequential morphisms and S-coalgebra morphisms coincide, hence $\mathsf{NSubseq}$ is a full subcategory of $\mathsf{Coalg}(S)$. This result is the basis for our slogan that *normalisation is coalgebraisation*. We also show that $\mathsf{NSubseq}$ has a final object, which must then also be final among all subsequential structures, since $\mathsf{NSubseq}$ is reflective. Choffrut's [37] results on the existence and properties of minimal subsequential transducers can be derived from the existence of this final object. The coalgebraic modelling of normalised structures also entails that in $\mathsf{NSubseq}$, state behaviour coincides with S-bisimilarity, and we describe how to adapt the minimisation algorithm for finite deterministic automata to finite normalised structures. Finally, we show that the subclasses of sequential and (partial) Mealy structures each are a category of coalgebras for some subfunctor of S and show that their final objects are substructures of the final normalised structure.

In the third part (section 4.5), we present an alternative coalgebraic modelling for subsequential transducers in which all states are final. We call such subsequential transducers (and their underlying structures) *step-by-step*. This coalgebraic representation is obtained by a structural transformation which corresponds with taking the differential of the behaviour. These so-called *differential representations* can be seen as sequential transducers which produce output in the free group $B^{(*)}$ rather than the free monoid B^*, and they can be modelled as coalgebras for a functor ${S_0}^{(*)}$. We can thus say that taking differentials is also a form of coalgebraisation. Moreover, like normalisation, taking differentials is functorial and a reflector. The practical interest of this coalgebraic characterisation is that it provides us with an alternative method for deciding

equivalence of step-by-step transducers which does not require normalisation. This method consists in computing state equivalence in the differential representation, and we compare it with the normalise-minimise approach at the end of the section. We also show that the minimal differential representation is the differential representation of the minimal, normalised representation.

The main contributions of this chapter are: (1) The classification of various subclasses of subsequential structures and transducers in terms of reflective subcategories and their coalgebraic or non-coalgebraic nature. In particular, the observation that normalisation and taking differentials are reflectors to categories of coalgebras. (2) A detailed description of the minimisation algorithms for normalised structures and differential representations. (3) A new approach for deciding equivalence of step-by-step transducers, which due to its local nature may be an efficient alternative to the existing method which requires a global normalisation procedure.

4.2 Preliminaries

4.2.1 Words, streams and functions.

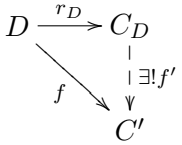
Let X and Y be sets. A (partial) function from X to Y is denoted by $f\colon X \dashrightarrow Y$. We will write $f\colon X \to Y$ when f is a total function from X to Y. The domain and range of $f\colon X \dashrightarrow Y$ are denoted $\mathrm{dom}(f)$ and $\mathrm{rng}(f)$, respectively. As is standard, we can view a partial function $f\colon X \dashrightarrow Y$ as a total function $f\colon X \to Y \cup \{\star\}$, where $\star$ is the *undefined value*, by letting $f(x) = \star$ for all $x \notin \mathrm{dom}(f)$. We will use the notation $\mathbf{1} = \{\star\}$ and write $\mathbf{1} + Y$ instead of $\{\star\} \cup Y$. For a function $f\colon X \dashrightarrow Y$, and subsets $C \subseteq X$ and $D \subseteq Y$, the direct f-image of C is denoted $f(C) = \{f(x) \in Y \mid x \in C\}$, the inverse f-image of D is $f^{-1}(D) = \{x \in X \mid f(x) \in D\}$, and the restriction of f to C is $f{\restriction_C}$.

The *free monoid over a set* X is the monoid $(X^*, \varepsilon, \mathbin{.})$ where X^* is the set of all (finite) words over X, ε is the empty word, and $u \mathbin{.} w$, or simply uw, denotes the concatenation of two words $u, w \in X^*$. For all $u, w \in X^*$, we write $u \preceq w$ if u is a prefix of w, i.e., there exists a $v \in X^*$ such that $w = uv$. The length of a word w is denoted by $|w|$ and we define $|\star| = 1$. If $f, g\colon X \to B^*$, then $f \mathbin{.} g\colon X \to B^*$ is the function defined by $(f \mathbin{.} g)(x) = f(x) \mathbin{.} g(x)$. The *free group over* X is denoted by $X^{(*)}$, and the *formal inverse of* $x \in X$ is written $\overline{x}$. For $w \in X^*$, the inverse of $w = x_1 x_2 \dots x_k$ is $\overline{w} = \overline{x_k} \dots \overline{x_2}\,\overline{x_1}$, and $\overline{\varepsilon} = \varepsilon$. We will apply concatenation and inverse to obtain prefixes and suffixes of words: If $w = uv \in X^*$, then $\overline{u} \mathbin{.} w = v$ and $w \mathbin{.} \overline{v} = u$. In the case u is not a prefix of $w \in X^*$, then $\overline{u} \mathbin{.} w$ is read as an element of $X^{(*)}$. For example, $\overline{aab} \mathbin{.} ab = \overline{b} \mathbin{.} \overline{a} \mathbin{.} \overline{a} \mathbin{.} ab = \overline{ab} \mathbin{.} b$. A subset $T \subseteq X^*$ is called *prefix-closed* if whenever $u \preceq w$ and $w \in T$ then $u \in T$. A partial function $f\colon X^* \dashrightarrow Y^*$ is *prefix-preserving* if $\mathrm{dom}(f)$ is prefix-closed, and for all $u, w \in \mathrm{dom}(f)$, if $u \preceq w$

then $f(u) \preceq f(w)$. For a set $S \subseteq X^*$ of words, we denote by $\mathrm{lcp}(S)$ the *longest common prefix* of words in S with the convention that $\mathrm{lcp}(\emptyset)$ is undefined.

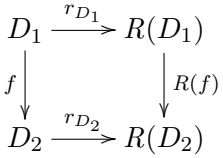
4.2.2 Reflective subcategories

We now recall the definition and some facts of reflective subcategories (see e.g. [2, 3, 91]). Let C be a subcategory of D, and D an object in D. A C-*reflection arrow* for D is a D-morphism $r_D \colon D \to C_D$ to some C-object C_D which has the following universal property. For any $C' \in \mathsf{C}$ and any D-morphism $f \colon D \to C'$ there is a unique C-morphism $f' \colon C_D \to C'$ such that $f = f' \circ r_D$. That is, the following diagram commutes.



The subcategory C of D is *reflective in* D if every D-object has a C-reflection arrow. As an example, we mention that in the category of deterministic automata (DA), the subcategory of minimal DA's is reflective (see [5, Chapter VI.1] and also [3, Example, 4.17]). More generally, for any Set-functor F, minimal F-coalgebras are reflective in $\mathsf{Coalg}(F)$ (cf. [50]).

An equivalent formulation of reflective subcategory is the following. A subcategory C of D is *reflective in* D if the embedding functor $E \colon \mathsf{C} \to \mathsf{D}$ has a left adjoint $R \colon \mathsf{D} \to \mathsf{C}$. This left adjoint R is called a *reflector*. Once a choice of reflection arrow has been made, the functor R can be defined by $R(D) = C_D$, and for $f \colon D_1 \to D_2$, $R(f)$ is the C-arrow determined by the following diagram:



Consequently, if C is reflective in D, then a final object C in C is also final in D, due to the bijection of Hom-sets: $\mathsf{D}(D, C) \cong \mathsf{C}(R(D), C)$. Namely, the unique morphism $h_{R(D)} \colon R(D) \to C$ corresponds to a unique morphism $h_D \colon D \to C$. Finally, we will use that reflectors compose, hence, if A is a reflective subcategory of B, and B is a reflective subcategory of C, then A is a reflective subcategory of C. These last two facts follow from more general results on adjoints. For example, if the embedding functor $E \colon \mathsf{C} \to \mathsf{D}$ has a left adjoint, it is itself a right adjoint. Now since right adjoints preserve limits, and final objects are limits, it follows that a final object in C is also final in D.

4.3 Subsequential structures and transducers

In this section we first review the basic definitions of subsequential transducers (cf. [36, 37]), and define subsequential structures and their morphisms. Next, we describe the subclasses of coaccessible, normalised and minimal subsequential structures. In particular, we characterise their morphisms and show that they form a sequence of nested, reflective subcategories in the category of all subsequential structures.

4.3.1 Basic definitions

Throughout this chapter, we assume we are given two (possibly infinite) sets A and B, which we refer to as the input and output alphabet, respectively. A subsequential transducer can be seen as a deterministic automaton (DA) which for every accepted input word from A^* produces an output word in B^*. This output is generated by outputting words on transitions as well as a terminal output word associated with final states. Moreover, the subsequential transducer may be equipped with an initial prefix which is a word that will be prefixed to the output generated from processing the input. Note that if the input word leads to a non-accepting state, then the output will be considered undefined. One should imagine that the machine starts with writing the initial prefix to some internal buffer, and as it reacts to the input, it concatenates the output from the transitions it has taken to the buffer word. When the end marker of the input word is encountered the machine checks whether it is in an accepting state; if yes, then it concatenates the terminal output of the current state to the buffer word, and outputs the result; otherwise it outputs the undefined value.

We mention that the initial prefix is needed in order to normalise subsequential transducers (see Subsection 4.3.3), but it does not add any expressivity, since a nontrivial prefix can always be "pushed into" the transducer by prefixing it to the output on all transitions leaving the initial state.

The input structure is assumed deterministic, but not total, that is, for each state and each input letter there is at most one transition available. For mathematical reasons, we allow the set of states to be empty, in which case the initial state and initial prefix are considered undefined.

4.3.1. Definition. A *subsequential structure* is a 4-tuple $\mathbb{S} = (Q, o, d, r)$ where Q is a set of states, $o\colon Q \to (A \dashrightarrow B^*)$ is an *output function*, $d\colon Q \to (A \dashrightarrow Q)$ is a *next-state function* and $r\colon Q \dashrightarrow B^*$ is a *terminal output function*. We require that for all $q \in Q$, $\mathrm{dom}(o(q)) = \mathrm{dom}(d(q)) =: \mathrm{supp}(q)$, called the support of q. The set of *final (or accepting)* states of $\mathbb{S}$ is $F := \mathrm{dom}(r)$. If $q \notin F$ then q called an *internal* state. The *underlying deterministic automaton* of $\mathbb{S}$ is the structure (Q, d, F). If A, B and Q are finite sets, then $\mathbb{S}$ is *finite*.

A *subsequential transducer* is a 6-tuple $\mathbb{T} = (Q, o, d, r, i, m)$ where (Q, o, d, r) is a subsequential structure, and if $Q \neq \emptyset$, $i \in Q$ is the *initial state*, and $m \in B^*$ is the *initial prefix*. In case $Q = \emptyset$, i and m are considered undefined, and $\mathbb{T}$ is called the empty transducer. $\lhd$

The usual notion of path in subsequential structures applies. A path is called *final* if it ends in a final state, and a state q is *coaccessible* if there exists a final path starting in q. The set of coaccessible states of a subsequential structure $\mathbb{S}$ (or transducer $\mathbb{T}$) will be denoted by $Coacc(\mathbb{S})$ (respectively, $Coacc(\mathbb{T})$). In a subsequential transducer a final path is *successful* if it starts in the initial state, and a state q is *accessible* (or reachable) if there is a path from the initial state to q. The set of accessible states of a subsequential transducer $\mathbb{T}$ are denoted by $Acc(\mathbb{T})$. A subsequential transducer is called *trimmed*, if all its states are accessible and coaccessible.

Let $\mathbb{S} = (Q, o, d, r)$ be a subsequential structure and $q \in Q$ a state. We extend the output and next-state functions at q to maps $o(q)\colon A^* \dashrightarrow B^*$ and $d(q)\colon A^* \dashrightarrow Q$ in the following standard manner. For $q \in Q$, $a \in A$ and $w \in A^*$, we define

$$\begin{array}{ll} d(q)(\varepsilon) = q, & d(q)(wa) = d(d(q)(w))(a), \\ o(q)(\varepsilon) = \varepsilon, & o(q)(wa) = o(q)(w) \centerdot o(d(q)(w))(a). \end{array}$$

with the proviso that the left-side is defined only if the right-side is. The set of words accepted from q in the underlying DA (Q, d, F) is denoted by $\mathrm{L}(q)$, i.e., $\mathrm{L}(q) = \{w \in A^* \mid d(q)(w) \in F\}$.

A subsequential transducer $\mathbb{T}$ computes a partial word function by its transformation of input words to output words. Similarly, given a subsequential structure $\mathbb{S}$ and a state q in $\mathbb{S}$, we can consider the partial word function computed by $\mathbb{S}$ when starting in q.

4.3.2. Definition. Given a subsequential structure $\mathbb{S} = (Q, o, d, r)$ and a state $q \in Q$, the *behaviour of q (in $\mathbb{S}$)* is the partial function $[\![q]\!]_{\mathbb{S}}\colon A^* \dashrightarrow B^*$ defined for all $w \in \mathrm{L}(q)$ by:

$$[\![q]\!]_{\mathbb{S}}(w) = o(q)(w) \centerdot r(d(q)(w)). \tag{4.1}$$

Given two subsequential structures $\mathbb{S}$ and $\mathbb{S}'$, two states q in $\mathbb{S}$ and q' in $\mathbb{S}'$ are *equivalent* if $[\![q]\!]_{\mathbb{S}} = [\![q']\!]_{\mathbb{S}'}$.

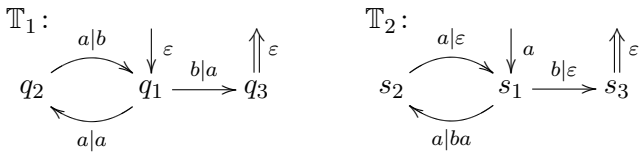
The *behaviour* of a subsequential transducer $\mathbb{T} = (\mathbb{S}, i, m)$ is the partial function $[\![\mathbb{T}]\!]\colon A^* \dashrightarrow B^*$ defined for all $w \in \mathrm{L}(i)$ by:

$$[\![\mathbb{T}]\!](w) = m \centerdot [\![i]\!]_{\mathbb{S}}(w). \tag{4.2}$$

We say that $\mathbb{T}$ *computes* $[\![\mathbb{T}]\!]$, and two subsequential transducers $\mathbb{T}_1$ and $\mathbb{T}_2$ are *equivalent* if $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$. A function $f\colon A^* \dashrightarrow B^*$ is called *subsequential*, if f is the behaviour of a finite subsequential transducer. $\lhd$

For notational simplicity we sometimes leave out the subscript from $[\![q]\!]_{\mathbb{S}}$ when $\mathbb{S}$ is clear from the context, or we use some appropriate indexing, for example, $[\![q]\!]_1$ instead of $[\![q]\!]_{\mathbb{S}_1}$, etc.

4.3.3. Example. Consider the subsequential transducers $\mathbb{T}_1$ and $\mathbb{T}_2$ depicted below. The initial state is marked by an incoming, sourceless arrow labelled with the initial prefix; a transition from a state q to a state q' on input letter a with output w is illustrated as an arrow from q to q' with label $a|w$; and final states are marked with an outgoing double-arrow labelled with the terminal output.



It is easy to see that $\mathbb{T}_1$ and $\mathbb{T}_2$ compute the same partial function $f\colon \{a,b\}^* \dashrightarrow \{a,b\}$, where $\mathrm{dom}(f) = \{(aa)^k b \mid k \in \omega\}$, and for all $k \in \omega$, $f((aa)^k b) = (ab)^k a$. Hence $\mathbb{T}_1$ and $\mathbb{T}_2$ are equivalent. ◁

The notion of morphism between trimmed subsequential transducers was introduced by Choffrut [37]. However, it applies also to subsequential transducers and structures in general, and in fact, under the assumption of coaccessibility, it can be somewhat simplified as we will see in subsection 4.3.2. Our current definition of subsequential morphisms is a slight variation on Choffrut's. See Remark 4.3.11 at the end of this subsection for more details on how the two relate to each other. Furthermore, we remark that our current definition differs from the one given earlier in [53], which turns out to be the correct notion for coaccessible structures, but not for subsequential structures in general, cf. subsection 4.3.2.

The below definition of subsequential morphisms may seem complicated at first sight, so we first try to give some intuitions. A morphism of subsequential transducers is a state mapping α which preserves behaviour. In order to guarantee the behaviours of q and $\alpha(q)$ have the same domains, it suffices to require that α respects the structure of the underlying DA's, when restricting to coaccessible states. In Example 4.3.3 above, the underlying DA's are isomorphic, and intuitively we would like the underlying DA-isomorphism to be a morphism from $\mathbb{T}_1$ to $\mathbb{T}_2$, since $\mathbb{T}_2$ is just like $\mathbb{T}_1$ except that, internally, $\mathbb{T}_2$ produces its output a bit faster than $\mathbb{T}_1$. Choffrut's observation in [36] was that, in general, it is possible to systematically shift some of the output letters "upstream" without changing the input-output behaviour. The definition of subsequential morphism essentially requires the existence of an output shift that makes the two subsequential transducers produce their output in a synchronised manner. We now give the formal definition.

4.3.4. Definition. Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be two subsequential structures. A function $\alpha\colon Q_1 \dashrightarrow Q_2$ is a *subsequential morphism* from $\mathbb{S}_1$ to $\mathbb{S}_2$ (notation: $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$), if there exists a function $\beta\colon Q_1 \to B^*$ such that the following conditions are satisfied for all $q \in Q_1$:

$$\begin{array}{ll}
\text{(next)} & \forall a \in A\colon \alpha(d_1(q)(a)) = d_2(\alpha(q))(a), \\
\text{(out)} & \forall a \in A\colon \text{ if } q, d_1(q)(a) \in \mathrm{dom}(\alpha) \\
 & \qquad \text{then } \beta(q) \centerdot o_2(\alpha(q))(a) = o_1(q)(a) \centerdot \beta(d_1(q)(a)), \\
\text{(acc)} & \alpha^{-1}(F_2) = F_1, \\
\text{(term-out)} & \forall q \in F_1\colon \beta(q) \centerdot r_2(\alpha(q)) = r_1(q).
\end{array}$$

We will use the notation $(\alpha, \beta)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ to say that α is a subsequential morphism from $\mathbb{S}_1$ to $\mathbb{S}_2$ with witnessing function β.

Given two subsequential transducers $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$, a subsequential morphism $(\alpha, \beta)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ is a *subsequential (transducer) morphism* from $\mathbb{T}_1$ to $\mathbb{T}_2$, if whenever $\mathbb{T}_1$ and $\mathbb{T}_2$ are not empty, α and β satisfy:

$$\begin{array}{ll}
\text{(init)} & \alpha(i_1) = i_2, \\
(\varepsilon\text{-in}) & m_2 = m_1 \centerdot \beta(i_1).
\end{array}$$

◁

Note that in Definition 4.3.4, the condition (next) should be read with a bit of care, due to the next-state function and α being partial maps. For example, if $a \notin \mathrm{supp}(\alpha(q))$, then (next) implies that either $a \notin \mathrm{supp}(q)$ or $d_1(q)(a) \notin \mathrm{dom}(\alpha)$. On the other hand, if $a \in \mathrm{supp}(\alpha(q))$, then $a \in \mathrm{supp}(q)$.

4.3.5. Example. We can now easily verify that in Example 4.3.3 the map $\alpha(q_j) = s_j$, $j \in \{1, 2, 3\}$, is a subsequential morphism from $\mathbb{T}_1$ to $\mathbb{T}_2$ by taking $\beta(q_1) = a$, $\beta(q_2) = ba$ and $\beta(q_3) = \varepsilon$. For example, to see that the (out) condition holds, we have

$$\begin{array}{lllllll}
\beta(q_1) \centerdot o_2(s_1)(a) & = & a \centerdot ba & = & a \centerdot ba & = & o_1(q_1)(a) \centerdot \beta(q_2) \\
\beta(q_2) \centerdot o_2(s_2)(a) & = & ba \centerdot \varepsilon & = & b \centerdot a & = & o_1(q_2)(a) \centerdot \beta(q_1) \\
\beta(q_1) \centerdot o_2(s_1)(b) & = & a \centerdot \varepsilon & = & a \centerdot \varepsilon & = & o_1(q_1)(b) \centerdot \beta(q_3)
\end{array}$$

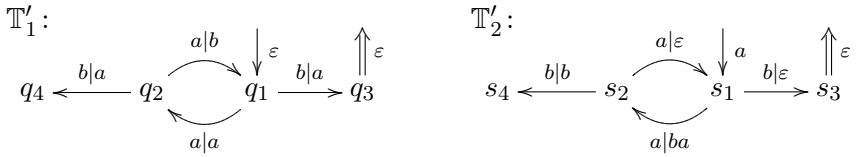
◁

We allow a subsequential morphism α to be partial in order to be able to ignore parts of the automaton structure which do not contribute to the behaviour. This is the reason why the condition (out) is restricted to states in $\mathrm{dom}(\alpha)$. The following lemma states that only states without behaviour may be ignored. Consequently, the empty map can only be a subsequential morphism if no states are coaccessible.

4.3.6. Lemma. *Let $\mathbb{S}_1$ and $\mathbb{S}_2$ be subsequential structures. If $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ is a subsequential morphism, then $Coacc(\mathbb{S}_1) \subseteq \mathrm{dom}(\alpha)$.*

PROOF. Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$, and assume $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$. By definition, $q \in \mathit{Coacc}(\mathbb{S}_1)$ iff there exists a $w \in A^*$ such that $d_1(q)(w) \in F_1$. We show by induction on the length of w that for all $q \in Q_1$, $d_1(q)(w) \in F_1$ implies $q \in \mathrm{dom}(\alpha)$. Base case: If $d_1(q)(\varepsilon) = q \in F_1$, then by (acc) we have $\alpha(q) \in F_2$, hence $q \in \mathrm{dom}(\alpha)$. Induction step: Let $w = av$ for $a \in A$ and $v \in A^*$, and assume $d_1(q)(av) \in F_1$. Since $d_1(q)(av) = d_1(d_1(q)(a))(v) \in F_1$, by applying the induction hypothesis to $d_1(q)(a)$ and v, we have $d_1(q)(a) \in \mathrm{dom}(\alpha)$. Now it follows from (next) that $d_2(\alpha(q))(a)$ is defined, hence, in particular, $q \in \mathrm{dom}(\alpha)$. QED

4.3.7. EXAMPLE. Consider the following two variations on the subsequential transducers of Example 4.3.3.



It can easily be confirmed that the map α given in Example 4.3.5 is also a subsequential morphism from $\mathbb{T}'_1$ to $\mathbb{T}'_2$ by defining β in q_4 to be any value. The states q_4 and s_4 are not coaccessible, and hence do not contribute to the behaviour of $\mathbb{T}'_1$ and $\mathbb{T}'_2$. In the next subsection (Lemma 4.3.13), we will see that if (α, β) is a subsequential morphism, then β is uniquely defined on all coaccessible states. In this example, this implies that q_4 cannot be in the domain of any subsequential morphism from $\mathbb{T}'_1$ to $\mathbb{T}'_2$, since there is no value for $\beta(q_4) \in B^*$ which would satisfy the (out) condition. ◁

As we mentioned already, a subsequential morphism α respects the underlying (partial) DA-structure. This implies that α preserves input language equivalence, i.e., domains of behaviours. The proof is standard, but we include it for completeness' sake.

4.3.8. LEMMA. *If $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ are subsequential structures and $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ a subsequential morphism, then for all $q \in \mathrm{dom}(\alpha) : \mathrm{L}(q) = \mathrm{L}(\alpha(q))$. In particular, $q \in \mathit{Coacc}(\mathbb{S}_1)$ iff $\alpha(q) \in \mathit{Coacc}(\mathbb{S}_2)$.*

PROOF. We prove by induction on the length of $w \in A^*$ that for all $q \in \mathrm{dom}(\alpha)$: $d_1(q)(w) \in F_1$ iff $d_2(\alpha(q))(w) \in F_2$. The base case follows from (acc). Now assume $w = av$ for $a \in A$ and $v \in A^*$. We have:

$$\begin{aligned} d_1(q)(av) \in F_1 &\iff d_1(d_1(q)(a))(v) \in F_1 \\ &\overset{\text{(IH)}}{\iff} d_2(\alpha(d_1(q)(a)))(v) \in F_2 \\ &\overset{\text{(next)}}{\iff} d_2(d_2(\alpha(q))(a))(v) \in F_2 \\ &\iff d_2(\alpha(q))(av) \in F_2. \end{aligned}$$

The last part of the lemma follows from the fact that $q \in \mathit{Coacc}(\mathbb{S}_1)$ iff $\mathrm{L}(q) \neq \emptyset$, similarly for $\alpha(q)$. QED

Subsequential morphisms preserve the behaviour of subsequential transducers, but not necessarily the behaviour of states, i.e., $(\alpha, \beta)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ *does not* imply that for all q in $\mathrm{dom}(\alpha)$, $[\![q]\!]_1 = [\![\alpha(q)]\!]_2$. This is, for example, the case in Example 4.3.3. Instead, given a subsequential morphism (α, β), the state behaviour of $\alpha(q)$ can be obtained from the state behaviour of q by explicit mention of $\beta(q)$.

4.3.9. Proposition. *Let $\mathbb{T} = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ be subsequential transducers. We have:*

1. *If $(\alpha, \beta)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$, then for all $q \in \mathrm{dom}(\alpha)$: $[\![q]\!]_1 = \beta(q) \centerdot [\![\alpha(q)]\!]_2$.*
2. *If $\alpha\colon \mathbb{T}_1 \dashrightarrow \mathbb{T}_2$, then $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$.*

Proof. *Item (1):* Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$, $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ and assume that $(\alpha, \beta)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$. From Lemma 4.3.8 it follows immediately that for all $q \in \mathrm{dom}(\alpha)$, $\mathrm{dom}([\![q]\!]_1) = \mathrm{dom}([\![\alpha(q)]\!]_2)$, hence, in particular, $[\![q]\!]_1$ is the empty map if and only if $[\![\alpha(q)]\!]_2$ is the empty map. We prove by induction on the length of $w \in \mathrm{dom}([\![q]\!]_1)$ that $[\![q]\!]_1(w) = \beta(q) \centerdot [\![\alpha(q)]\!]_2(w)$. Base case:

$$[\![q]\!]_1(\varepsilon) = r_1(q) \overset{\text{(term-out)}}{=} \beta(q) \centerdot r_2(\alpha(q)) = \beta(q) \centerdot [\![\alpha(q)]\!]_2.$$

Induction step: Let $w = av \in \mathrm{dom}([\![q]\!]_1)$ where $a \in A$ and $v \in A^*$. Note that this implies that q and $d(q)(a)$ are in $\mathrm{dom}(\alpha)$. We have:

$$\begin{array}{rcl}
[\![q]\!]_1(av) & = & o_1(q)(a) \centerdot [\![d_1(q)(a)]\!]_1(v) \\
& \overset{\text{(IH)}}{=} & o_1(q)(a) \centerdot \beta(d_1(q)(a)) \centerdot [\![\alpha(d_1(q)(a))]\!]_2(v) \\
& \overset{\text{(out)}}{=} & \beta(q) \centerdot o_2(\alpha(q))(a) \centerdot [\![\alpha(d_1(q)(a))]\!]_2(v) \\
& \overset{\text{(next)}}{=} & \beta(q) \centerdot o_2(\alpha(q))(a) \centerdot [\![d_2(\alpha(q))(a)]\!]_2(v) \\
& = & \beta(q) \centerdot [\![\alpha(q)]\!](av).
\end{array}$$

Item (2): Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_2)$, $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ and $(\alpha, \beta)\colon \mathbb{T}_1 \dashrightarrow \mathbb{T}_2$ be a subsequential transducer morphism. By definition, for $w \in A^*$: $[\![\mathbb{T}_1]\!](w) = m_1 \centerdot [\![i_1]\!]_1(w)$. From item (1) and (init), we get $[\![\mathbb{T}_1]\!](w) = m_1 \centerdot \beta(i_1) \centerdot [\![i_2]\!]_2(w)$, and finally from ($\varepsilon$-in), $[\![\mathbb{T}_1]\!](w) = m_2 \centerdot [\![i_2]\!]_2(w) = [\![\mathbb{T}_2]\!](w)$. QED

Subsequential morphisms can be composed as described in the following lemma (the straightforward proof is omitted).

4.3.10. Lemma. *If $(\alpha_1, \beta_1)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$, and $(\alpha_2, \beta_2)\colon \mathbb{S}_2 \dashrightarrow \mathbb{S}_3$, are subsequential morphisms, then $(\alpha_2 \circ \alpha_1, \beta_1 \centerdot (\beta_2 \circ \alpha_1))\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_3$.*

For a subsequential structure $\mathbb{S} = (Q, o, d, r)$, we define the identity morphism $id_{\mathbb{S}}$ on $\mathbb{S}$ to be the identity map id_Q on the state set Q. It is easily seen that id_Q is a subsequential morphism from $\mathbb{S}$ to $\mathbb{S}$ by taking $\beta = \varepsilon$ (the constant function equal to ε everywhere). Hence subsequential structures and subsequential morphisms form a category Subseq. Similarly, for a subsequential transducer $\mathbb{T}$ with state set Q, the identity morphism on $\mathbb{T}$ is defined as $id_{\mathbb{T}} := id_Q$, and subsequential transducers and subsequential transducer morphisms form a category $\mathsf{SubseqTra}$. Note that although subsequential morphisms allow a non-trivial output shift β, an isomorphism in Subseq is a subsequential morphism $(\alpha, \beta)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ for which α is a bijection between the state sets and $\beta\restriction_{Coacc(\mathbb{S}_1)} = \varepsilon$. To see this, suppose $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$, $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$, $(\alpha_1, \beta_1)\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ and $(\alpha_2, \beta_2)\colon \mathbb{S}_2 \dashrightarrow \mathbb{S}_1$ such that $\alpha_2 \circ \alpha_1 = id_{\mathbb{S}_1}$ and $\alpha_1 \circ \alpha_2 = id_{\mathbb{S}_2}$. Clearly, $\alpha_1\colon Q_1 \to Q_2$ is a bijection. From condition (term-out) we get that for all $q \in F_1$: $\beta_1(q) = r_1(q) \centerdot \overline{r_2(\alpha_1(q))}$ and $\beta_2(\alpha_1(q)) = r_2(\alpha_1(q)) \centerdot \overline{r_1(q)}$. This means that $\beta_1(q) = \overline{\beta_2(\alpha(q))}$. Since β_1 and β_2 take values in B^*, we must have that $\beta_1(q) = \beta_2(\alpha_1(q)) = \varepsilon$ for all $q \in F_1$. We can extend this argument to show that for all $q \in Coacc(\mathbb{S}_1)$, $\beta_1(q) = \beta_2(\alpha_1(q)) = \varepsilon$ by induction on the distance of a state q to F_1 (as in the proof of Lemma 4.3.8) and condition (out).

4.3.11. Remark. Definition 4.3.4 is adapted from Choffrut [37] who defines morphisms of trimmed subsequential transducers. We remark the following:

(i) Choffrut allows β to take values in $B^* \cup \overline{B^*}$, where $\overline{B^*} = \{\overline{w} \mid w \in B^*\} \subseteq B^{(*)}$. This slightly more general definition allows morphisms to exist from $\mathbb{T}_1$ to $\mathbb{T}_2$, also if $\mathbb{T}_2$ sometimes produces its output slower than $\mathbb{T}_1$. We find this an unnecessary generalisation, since it is not needed to prove the existence of a minimal subsequential transducer (cf. Definition 4.3.28 and Corollary 4.4.8).

(ii) We note that Choffrut also defines a subsequential morphism as a partial map α, however, since all states in a trimmed subsequential transducer are coaccessible, it follows from conditions (acc) and (next) (cf. Lemma 4.3.6) that α must in fact be a total function. ◁

More results on subsequential transducers and subsequential functions can be found in [18, 23, 29, 36, 37, 122], including methods of determinisation (Béal & Carton [18]), and a characterisation of subsequential functions (Choffrut [36]) which generalises the Ginsburg-Rose theorem for sequential functions.

4.3.2 Coaccessible structures and trimmed transducers

We call a subsequential structure (or transducer) coaccessible, if all its states are coaccessible. We denote by $\mathsf{CSubseq}$ the full subcategory of Subseq consisting of coaccessible subsequential structures; similarly, $\mathsf{CSubseqTra}$ is the full subcategory of $\mathsf{SubseqTra}$ consisting of coaccessible subsequential transducers.

It is well-known that given a finite (partial) deterministic automaton, one can obtain the coaccessible part by computing the states that are backwards reachable from the final states, and the same, of course, holds for subsequential structures, since coaccessibility is a property defined with respect to the underlying DA.

4.3.12. Definition (Coaccessible part). Let $\mathbb{S} = (Q, o, d, r)$ be a subsequential structure. We define the *coaccessible part of* $\mathbb{S}$ as $\mathcal{C}(\mathbb{S}) := (Q', o', d', r)$ where $Q' = \mathit{Coacc}(\mathbb{S})$ and o' and d' are the restrictions of o and d to Q', i.e., for all $q \in Q'$ and $a \in A$, $o'(q)(a) = o(q)(a)$ and $d'(q)(a) = d(q)(a)$, if $d(q)(a) \in Q'$, otherwise $o'(q)(a)$ and $d'(q)(a)$ are undefined. ◁

In the previous subsection, we observed in Lemma 4.3.6 that a subsequential morphism must be defined on all coaccessible states, hence the morphisms in $\mathsf{CSubseq}$ are total maps. We now show that Definition 4.3.4 ensures that the witnessing output shift function β is uniquely defined on coaccessible states.

4.3.13. Lemma. *Let* $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ *and* $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ *be subsequential structures and* $\alpha \colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$ *a subsequential morphism. If* $\beta : Q_1 \to B^*$ *and* $\beta' : Q_1 \to B^*$ *are both witnessing functions for* α*, then* $\beta{\restriction}_{\mathit{Coacc}(\mathbb{S}_1)} = \beta'{\restriction}_{\mathit{Coacc}(\mathbb{S}_1)}$.

Proof. Let $q \in \mathit{Coacc}(\mathbb{S}_1)$. Then there exists some $w \in A^*$ such that $d_1(q)(w) \in F_1$ and w is of minimal length. We show by induction on the length of w that $\beta(q) = \beta'(q)$. If $w = \varepsilon$, then $\beta(q) = \beta'(q)$ follows from (term-out). In the case $w = av$ for $v \in A^*$ and $a \in A$, we have that $d_1(q)(a)$ is also coaccessible, and v is a word of minimal length labelling a final path starting in $q_a = d_1(q)(a)$, hence by induction hypothesis (IH) and (out) it now follows that

$$\begin{aligned} \beta(q) &\overset{\text{(out)}}{=} o_1(q)(a) \centerdot \beta(q_a) \centerdot \overline{o_2(\alpha(q))(a)} \\ &\overset{\text{(IH)}}{=} o_1(q)(a) \centerdot \beta'(q_a) \centerdot \overline{o_2(\alpha(q))(a)} \overset{\text{(out)}}{=} \beta'(q). \end{aligned}$$ QED

In the next proposition, we characterise the subsequential morphisms between coaccessible structures.

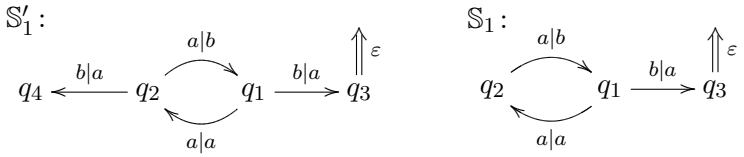
4.3.14. Proposition. *Let* $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ *and* $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ *be coaccessible subsequential structures. A function* $\alpha \colon Q_1 \dashrightarrow Q_2$ *is a subsequential morphism from* $\mathbb{S}_1$ *to* $\mathbb{S}_2$*, if and only if* $\mathrm{dom}(\alpha) = Q_1$ *and there exists a function* $\beta \colon Q_1 \to B^*$ *such that the following conditions are satisfied for all* $q \in Q_1$*:*

(supp)	$\mathrm{supp}(q) = \mathrm{supp}(\alpha(q))$,
$(\text{next})_{\mathsf{C}}$	$\forall a \in \mathrm{supp}(q) \colon \alpha(d_1(q)(a)) = d_2(\alpha(q))(a)$,
$(\text{out})_{\mathsf{C}}$	$\forall a \in \mathrm{supp}(q) \colon \beta(q) \centerdot o_2(\alpha(q))(a) = o_1(q)(a) \centerdot \beta(d_1(q)(a))$,
(acc)	$\alpha^{-1}(F_2) = F_1$,
(term-out)	$\forall q \in F_1 \colon \beta(q) \centerdot r_2(\alpha(q)) = r_1(q)$.

PROOF. If α satisfies the conditions, α is clearly a subsequential morphism. Conversely, if α is a subsequential morphism, then it follows from Lemma 4.3.6 that α is total, and when α is total, condition (next) ensures that $\mathrm{supp}(q) = \mathrm{supp}(\alpha(q))$ for all $q \in Q_1$, and the output and next-state conditions need to be checked exactly for input letters in the support. QED

We will use the notation $\alpha\colon \mathbb{S}_1 \to \mathbb{S}_2$ rather than $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$, when $\mathbb{S}_1$ is coaccessible, in order to emphasise that α is a total map.

4.3.15. EXAMPLE. Consider the subsequential structures $\mathbb{S}_1$ and $\mathbb{S}'_1$ underlying $\mathbb{T}_1$ from Example 4.3.3 and $\mathbb{T}'_1$ from Example 4.3.7, as illustrated below.



Clearly, $\mathbb{S}_1 = \mathcal{C}(\mathbb{S}'_1)$, and it is easy to see that $(id_{Coacc(\mathbb{S}'_1)}, \varepsilon)\colon \mathbb{S}'_1 \dashrightarrow \mathbb{S}_1$ in Subseq. ◁

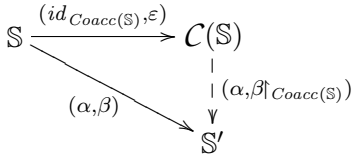
The morphism in the example above is, in fact, a reflection arrow, and we now show that coaccessible structures form a reflective subcategory of Subseq. This result confirms that our definition of subsequential morphisms is the correct one. It is also an argument for saying that the right way of thinking about subsequential structures is in terms of their coaccessible part. We will see in the next subsection that this statement can be sharpened by considering normalised subsequential structures.

4.3.16. THEOREM. *Let $\mathbb{S}$ be a subsequential structure. We have:*

$$(id_{Coacc(\mathbb{S})}, \varepsilon)\colon \mathbb{S} \dashrightarrow \mathcal{C}(\mathbb{S}) \text{ is a } \mathsf{CSubseq}\text{-reflection arrow for } \mathbb{S}.$$

It follows that $\mathsf{CSubseq}$ *is a reflective subcategory of* Subseq*, and the map $\mathcal{C}$ is a functor $\mathcal{C}\colon \mathsf{Subseq} \to \mathsf{CSubseq}$ by defining $\mathcal{C}(\alpha) = \alpha{\restriction}_{Coacc(\mathbb{S})}$ for $\alpha\colon \mathbb{S}_1 \dashrightarrow \mathbb{S}_2$.*

PROOF. Let $\mathbb{S}$ be a subsequential structure. It is straightforward to check that the map $id_{Coacc(\mathbb{S})}$ is a subsequential morphism from $\mathbb{S}$ to $\mathcal{C}(\mathbb{S})$. It remains to show that $id_{Coacc(\mathbb{S})}$ has the desired universal property, that is, for any subsequential morphism $(\alpha, \beta)\colon \mathbb{S} \dashrightarrow \mathbb{S}'$ in Subseq where $\mathbb{S}'$ is in $\mathsf{CSubseq}$, there is a unique morphism $\alpha'\colon \mathcal{C}(\mathbb{S}) \to \mathbb{S}'$ in $\mathsf{CSubseq}$ such that $\alpha = \alpha' \circ id_{Coacc(\mathbb{S})}$. We claim that $\alpha' = \alpha$ with witnessing function $\beta' = \beta{\restriction}_{Coacc(\mathbb{S})}$ as illustrated in the following diagram.



To prove our claim, we just have to note that by Lemmas 4.3.6 and 4.3.8 and the assumption that $\mathbb{S}'$ is coaccessible, it follows that $\mathrm{dom}(\alpha) = \mathit{Coacc}(\mathbb{S})$, hence $\alpha = \alpha{\restriction}_{\mathit{Coacc}(\mathbb{S})} = \alpha \circ \mathit{id}_{\mathit{Coacc}(\mathbb{S})}$, and α is clearly unique. It is also easy to see that $\beta' = \beta{\restriction}_{\mathit{Coacc}(\mathbb{S})}$ witnesses the fact that $\alpha\colon \mathcal{C}(\mathbb{S}) \to \mathbb{S}'$ is a morphism. Moreover, β' is unique due to Lemma 4.3.13. QED

Subsequential transducers are often assumed to be trimmed (cf. [37]), and we now look closer at the operations involved in *trimming* a transducer. First we note that for subsequential transducers, taking the coaccessible part is not always a well-defined operation. The reason is that given a subsequential transducer in which the initial state is not coaccessible and the set of final states is not empty, restricting to the coacessible part of the underlying structure will result in an object where the initial state is not an element of the non-empty state set. Such objects are not subsequential transducers by our definition. Still, the category $\mathsf{CSubseqTra}$ is well-defined, only $\mathcal{C}$ is not a functor from $\mathsf{SubseqTra}$ to $\mathsf{CSubseqTra}$. Before we can take the coaccessible part, we must first make $\mathbb{T}$ accessible.

4.3.17. Definition (Accessible part). Let $\mathbb{T} = (Q, o, d, r, i, m)$ be a subsequential transducer. We define $\mathcal{A}(\mathbb{T}) = (\mathit{Acc}(\mathbb{T}), o', d', r', i, m)$ where o', d' and r' are the restrictions of o, d and r to $\mathit{Acc}(\mathbb{T})$. $\mathcal{A}(\mathbb{T})$ is called the *accessible part* of $\mathbb{T}$, and $\mathbb{T}$ is called *accessible* if $\mathbb{T} = \mathcal{A}(\mathbb{T})$. If $\mathbb{T} = (\mathbb{S}, i, m)$ is accessible, then we define $\mathcal{C}(\mathbb{T}) = (\mathcal{C}(\mathbb{S}), i, m)$ with i and m undefined if $\mathit{Coacc}(\mathbb{S}) = \emptyset$. ◁

$\mathcal{A}(\mathbb{T})$ is clearly a subsequential transducer, and for a finite subsequential transducer $\mathbb{T}$, $\mathit{Acc}(\mathbb{T})$ can be effectively computed as the states that are reachable from the initial state. It is now also clear that given an arbitrary subsequential transducer $\mathbb{T}$, $\mathcal{C}(\mathcal{A}(\mathbb{T}))$ is trimmed. Let $\mathsf{ASubseqTra}$ and $\mathsf{TSubseqTra}$ denote the full subcategories of $\mathsf{SubseqTra}$ consisting of accessible and trimmed subsequential transducers, respectively.

The notions of accessibility and coaccessibility are properties of the underlying deterministic automaton, and the following proposition is therefore easily adapted to a statement about (partial) deterministic automata.

4.3.18. Proposition. *We have:*

1. *For all* $\mathbb{T} \in \mathsf{SubseqTra}$, $\mathit{id}_{\mathit{Acc}(\mathbb{T})}\colon \mathcal{A}(\mathbb{T}) \dashrightarrow \mathbb{T}$ *is a subsequential morphism.*
2. *For all* $\mathbb{T} \in \mathsf{ASubseqTra}$, $\mathit{id}_{\mathit{Coacc}(\mathbb{T})}\colon \mathbb{T} \dashrightarrow \mathcal{C}(\mathbb{T})$ *is a* $\mathsf{TSubseqTra}$*-reflection arrow.*

Proof. We only provide a sketch. For item (1), let $\mathbb{T} = (Q, o, d, r, i, m)$ and $\mathcal{A}(\mathbb{T}) = (\mathit{Acc}(\mathbb{T}), o', d', r', i, m)$. To see that $\mathit{id}_{\mathit{Acc}(\mathbb{T})}$ satisfies (next), let

$q \in Acc(\mathbb{T})$ and $a \in A$. We then have $id_{Acc(\mathbb{T})}(d'(q)(a)) = \star$ iff $a \notin \mathrm{supp}(q)$ iff $d(id_{Acc(\mathbb{T})}(q))(a) = \star$, since q is accessible. Condition (acc) holds since $F' := \mathrm{dom}(r') = F \cap Acc(\mathbb{T}) = id^{-1}_{Acc(\mathbb{T})}(F)$. The other conditions from Definition 4.3.4 are checked as easily. Item (2) can be proved along the same lines as Theorem 4.3.16. QED

Since behaviour is invariant under subsequential transducer morphisms, we have an easy corollary.

4.3.19. Corollary. *For any subsequential transducer $\mathbb{T}$, $\mathcal{C}(\mathcal{A}(\mathbb{T}))$ is equivalent with $\mathbb{T}$.*

In fact, it is also possible to show that for any subsequential transducer $\mathbb{T}$, $id_{Acc(\mathbb{T})} \colon \mathcal{A}(\mathbb{T}) \dashrightarrow \mathbb{T}$ is an $\mathsf{ASubseqTra}$-coreflection arrow for $\mathbb{T}$. A coreflection arrow is the dual notion of a reflection arrow. Hence trimming a transducer can be seen as a composition of a coreflection with a reflection. We leave it to the interested reader to verify this claim, since it will not play any role in the rest of the chapter.

4.3.3 Normalised subsequential structures

As we have seen in the previous subsection, considering coaccessible subsequential structures allows us to work with morphisms as total maps and unique witnessing functions. Still, in spite of this conceptual simplification, checking whether a morphism exists between two coaccessible structures, or whether two subsequential transducers are equivalent, is complicated by checking for the existence of a witnessing output shift function β. In this subsection, we will see that this problem is eliminated by considering normalised subsequential structures and transducers. Informally stated, a normalised subsequential transducer produces its output at maximal speed. Consequently, morphisms between normalised subsequential transducers can only have $\beta = \varepsilon$ as witnessing function.

The definition of normalised subsequential transducers goes back to Choffrut [36] who showed that any finite subsequential transducer can be transformed into an equivalent normalised one. Here we formulate Choffrut's results for coaccessible subsequential structures, and we show that normalised subsequential structures and transducers form reflective subcategories of $\mathsf{CSubseq}$ and $\mathsf{CSubseqTra}$, respectively.

4.3.20. Definition. Let $\mathbb{S} = (Q, o, d, r)$ be a coaccessible subsequential structure, and $q \in Q$. We define a function $\hat{\beta}_{\mathbb{S}} \colon Q \to B^*$ by

$$\hat{\beta}_{\mathbb{S}}(q) = \mathrm{lcp}(\{o(q)(w) \centerdot r(d(q)(w)) \mid w \in \mathrm{L}(q)\}). \tag{4.3}$$

That is, $\hat{\beta}_{\mathbb{S}}(q)$ is the longest common prefix over all output words on final paths starting in q. A state $q \in Q$ *is normalised* if $\hat{\beta}_{\mathbb{S}}(q) = \varepsilon$. A subsequential structure $\mathbb{S}$ is *normalised*, if all states in $\mathbb{S}$ are normalised, and a subsequential transducer $\mathbb{T}$ is *normalised* if its underlying subsequential structure is normalised. ◁

If $\mathbb{T} = (\mathbb{S}, i, m)$, then will use the notation $\hat{\beta}_{\mathbb{T}} = \hat{\beta}_{\mathbb{S}}$, or we may leave out the subscript altogether if no confusion can arise. Let $\mathsf{NSubseq}$ be the full subcategory of $\mathsf{CSubseq}$ (and Subseq) consisting of normalised subsequential structures and subsequential morphisms. Similarly, $\mathsf{NSubseqTra}$ is the full subcategory of $\mathsf{CSubseqTra}$ consisting of normalised subsequential transducers.

The meaning of $\hat{\beta}$ can be explained informally as follows. Suppose a subsequential transducer $\mathbb{T}$ is processing an input word $w = vu$, and after reading v, the output produced so far is $x \in A^*$ and the current state is q. Now $\hat{\beta}(q)$ gives us the longest word which will be output by $\mathbb{T}$ in the remainder of the computation, no matter what u is (assuming $w \in \mathrm{dom}([\![\mathbb{T}]\!])$). But this means that the output of $\hat{\beta}(q)$ is unnecessarily delayed while waiting for $\mathbb{T}$ to read the next input letter. Normalising a subsequential transducer consists in changing the output functions such that there is no delayed output anywhere.

4.3.21. Definition. Let $\mathbb{T} = (Q, o, d, r, i, m)$ be a coaccessible subsequential transducer. The *normalisation of* $\mathbb{T}$ is the subsequential transducer $\mathcal{N}(\mathbb{T}) = (Q, o', d, r', i, m')$ where for all $q \in Q$, and all $a \in A$:

$$m' = m \centerdot \hat{\beta}(i), \quad o'(q)(a) = \overline{\hat{\beta}(q)} \centerdot o(q)(a) \centerdot \hat{\beta}(d(q)(a)), \quad r'(q) = \overline{\hat{\beta}(q)} \centerdot r(q) \tag{4.4}$$

Similarly, if $\mathbb{S} = (Q, o, d, r)$ is a subsequential structure, then the *normalisation of* $\mathbb{S}$ is $\mathcal{N}(\mathbb{S}) = (Q, o', d, r')$, where o' and r' are defined as in (4.4). ◁

Note that in Definition 4.3.21, $o'(q)(a)$ and $r'(q)$ are in B^* for all states q and $a \in A$, since $\hat{\beta}(q)$ is a prefix of both $o(q)(a) \centerdot \hat{\beta}(d(q)(a))$ and $r(q)$.

Before we characterise subsequential morphisms between normalised structures, we note that from Proposition 4.3.9 it follows that for any subsequential morphism $(\alpha, \beta) \colon \mathbb{S}_1 \to \mathbb{S}_2$ in $\mathsf{CSubseq}$, and all states q in $\mathbb{S}_1$:

$$\hat{\beta}_1(q) = \beta(q) \centerdot \hat{\beta}_2(\alpha(q)). \tag{4.5}$$

From (4.5) it follows immediately that:

4.3.22. Lemma. *Let* $(\alpha, \beta) \colon \mathbb{S}_1 \to \mathbb{S}_2$ *be a* $\mathsf{CSubseq}$*-morphism.*

1. *If* $\mathbb{S}_2$ *is normalised, then* $\beta = \hat{\beta}_1$.
2. *If* $\mathbb{S}_1$ *and* $\mathbb{S}_2$ *are normalised, then* $\beta = \varepsilon$.

We can now prove that subsequential morphisms between normalised structures are very much like morphisms between deterministic automata or Mealy machines.

4.3.23. Proposition. *Let* $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ *and* $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ *be normalised subsequential structures. A function* $\alpha : Q_1 \to Q_2$ *is a subsequential morphism if and only if* $(\alpha, \varepsilon) \colon \mathbb{S}_1 \to \mathbb{S}_2$*, i.e., for all* $q \in Q_1$*:*

$$\begin{array}{ll}
\text{(supp)} & \mathrm{supp}(q) = \mathrm{supp}(\alpha(q)), \\
(\text{next})_{\mathsf{C}} & \forall a \in \mathrm{supp}(q) \colon \alpha(d_1(q)(a)) = d_2(\alpha(q))(a), \\
(\text{out})_{\mathsf{N}} & \forall a \in \mathrm{supp}(q) \colon o_1(q)(a) = o_2(\alpha(q))(a), \\
\text{(acc)} & \alpha^{-1}(F_2) = F_1, \\
(\text{term-out})_{\mathsf{N}} & \forall q \in \mathrm{dom}(r_1) \colon r_1(q) = r_2(\alpha(q)).
\end{array}$$

Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ *and* $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ *be normalised subsequential transducers. A subsequential morphism* $\alpha \colon \mathbb{S}_1 \to \mathbb{S}_2$ *is a subsequential transducer morphism* $\alpha \colon \mathbb{T}_1 \to \mathbb{T}_2$ *if and only if whenever* $\mathbb{T}_1$ *and* $\mathbb{T}_2$ *are not empty*

$$\begin{array}{ll}
\text{(init)} & \alpha(i_1) = i_2, \text{ and} \\
(\varepsilon\text{-in})_{\mathsf{N}} & m_2 = m_1.
\end{array}$$

Proof. First consider the characterisation of subsequential morphisms between normalised structures. If $\alpha \colon \mathbb{S}_1 \to \mathbb{S}_2$, then by Lemma 4.3.22(2) the (unique) witnessing function is $\beta = \varepsilon$. The other direction is clear. The characterisation of morphisms between normalised transducers follows easily from the result for structures. QED

An easy consequence of Lemma 4.3.22 is that subsequential morphisms between normalised structures preserve state behaviour.

4.3.24. Proposition. *If* $\alpha \colon \mathbb{S}_1 \to \mathbb{S}_2$ *is a subsequential morphism in* NSubseq*, then for all states* q *in* $\mathbb{S}_1$*:* $[\![q]\!]_1 = [\![\alpha(q)]\!]_2$.

Proof. From Proposition 4.3.9, we have for all q in $\mathbb{S}_1$ that $[\![q]\!]_1 = \beta(q) \centerdot [\![\alpha(q)]\!]_2$. Since $\mathbb{S}_2$ is normalised, it follows from Lemma 4.3.22(2) that $\beta(q) = \varepsilon$, hence $[\![q]\!]_1 = [\![\alpha(q)]\!]_2$. QED

We now show that normalisation is a reflector. This result takes the argument from the previous subsection for coaccessible structures and transducers a step further, so that we now can say that the right way of thinking about subsequential structures and transducers is in terms of their normalisation.

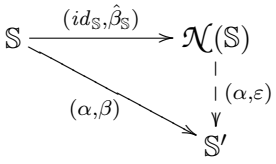
4.3.25. THEOREM. *Let* $\mathbb{S}$ *be a coaccessible subsequential structure and* $\mathbb{T} = (\mathbb{S}, i, m)$ *a coaccessible subsequential transducer. We have:*

1. $(id_{\mathbb{S}}, \hat{\beta}_{\mathbb{S}})\colon \mathbb{S} \dashrightarrow \mathcal{N}(\mathbb{S})$ *is an* NSubseq*-reflection arrow for* $\mathbb{S}$.

2. $(id_{\mathbb{T}}, \hat{\beta}_{\mathbb{T}})\colon \mathbb{T} \dashrightarrow \mathcal{N}(\mathbb{T})$ *is an* NSubseqTra*-reflection arrow for* $\mathbb{T}$.

It follows that NSubseq *is a reflective subcategory of* CSubseq*, and* NSubseqTra *is a reflective subcategory of* CSubseqTra*. Moreover, the map* $\mathcal{N}$ *is a functor* $\mathcal{N}\colon$ CSubseq $\to$ NSubseq *and* $\mathcal{N}\colon$ CSubseqTra $\to$ NSubseqTra *by defining* $\mathcal{N}(\alpha) = \alpha$ *for a subsequential morphism* α *in* CSubseq *or* CSubseqTra .

PROOF. *Item (1):* Let $\mathbb{S}$ be an object in CSubseq. It is straightforward to check that $(id_{\mathbb{S}}, \hat{\beta}_{\mathbb{S}})\colon \mathbb{S} \to \mathcal{N}(\mathbb{S})$ is a subsequential morphism by using the characterisation of morphisms in CSubseq (Proposition 4.3.14) and the definition of $\mathcal{N}(\mathbb{S})$ (Definition 4.3.21). Now it is also easy to see that for any normalised $\mathbb{S}' \in$ NSubseq and any CSubseq-morphism $(\alpha, \beta)\colon \mathbb{S} \to \mathbb{S}'$, the unique NSubseq-morphism $\alpha'\colon \mathcal{N}(\mathbb{S}) \to \mathbb{S}'$ such that $\alpha' \circ id_{\mathbb{S}} = \alpha$ is just $\alpha' = \alpha$. From Lemma 4.3.22(2) it follows that the witnessing function β' for $\alpha\colon \mathcal{N}(\mathbb{S}) \to \mathbb{S}'$ is just $\beta' = \varepsilon$, and that the diagram on the right commutes.



Item (2): Let $\mathbb{T} = (\mathbb{S}, i, m,)$ be in CSubseqTra. The map $id_{\mathbb{T}}\colon \mathbb{T} \to \mathcal{N}(\mathbb{T})$ is a subsequential morphism more or less by definition of $\mathcal{N}(\mathbb{T})$. Now suppose $(\alpha, \beta)\colon \mathbb{T} \to \mathbb{T}'$ is a subsequential morphism for some $\mathbb{T}' = (\mathbb{S}', i', m')$ in NSubseqTra. It follows that, in particular, $(\alpha, \beta)\colon \mathbb{S} \to \mathbb{S}'$, hence from item (1), we have that $(\alpha, \varepsilon)\colon \mathcal{N}(\mathbb{S}) \to \mathbb{S}'$ is the unique subsequential morphism such that $\alpha = \alpha \circ id_{\mathbb{S}}$. Again, α is a transducer morphism from $\mathcal{N}(\mathbb{T})$ to $\mathbb{T}'$, since from (init) we get $m' = m \centerdot \beta(i)$, and using Lemma 4.3.22(1), $m' = m \centerdot \hat{\beta}_{\mathbb{S}}(i)$, hence $\mathcal{N}(\mathbb{T})$ and $\mathbb{T}'$ have the same prefix. The uniqueness of α follows from the uniqueness of α on the underying structures. QED

4.3.26. COROLLARY. *For all* $\mathbb{T}$ *in* CSubseqTra*, we have:* $[\![\mathbb{T}]\!] = [\![\mathcal{N}(\mathbb{T})]\!]$.

PROOF. Follows from Theorem 4.3.25(2), and the fact that subsequential morphisms preserve transducer behaviour. QED

Choffrut surveys in [37] a number of different algorithms for computing $\hat{\beta}$. One of these algorithms is by Béal & Carton [17] who report that for a normalised $\mathbb{S}$, $\hat{\beta}$ can be computed in time $O((\|\hat{\beta}\| + 1)M)$, where $\|\hat{\beta}\|$ is the maximal length of $\hat{\beta}(q)$ for all states q in $\mathbb{S}$, and M is the number of transitions in $\mathbb{S}$.

4.3.4 Minimal subsequential transducers

A subsequential structure $\mathbb{S}$ is called *minimal*, if $\mathbb{S}$ is normalised and no two states in $\mathbb{S}$ are equivalent. Similarly, a subsequential transducer is minimal if its underlying structure is minimal. Choffrut showed in [37] that for any function $f\colon A^* \dashrightarrow B^*$, there exists a minimal (but possibly infinite) subsequential transducer $\mathbb{T}_f$ with behaviour f such that for all trimmed subsequential transducers $\mathbb{T}$ which also compute f, there is a unique subsequential morphism $\alpha\colon \mathbb{T} \to \mathbb{T}_f$. This result strongly suggests the existence of a final normalised subsequential structure, and in the next section we will prove that indeed the existence and properties of $\mathbb{T}_f$ follow from finality, (Corollary 4.4.8). For now we just give the definition of $\mathbb{T}_f$ which is based on the notion of maximal output and derivative (also called the residual) of word functions.

4.3.27. Definition. Let $f\colon A^* \dashrightarrow B^*$ and $w \in A^*$. The *maximal output of f on input w* is given by

$$f[w] \quad := \quad \mathrm{lcp}(f(wA^*)) = \mathrm{lcp}(\{f(wu) \mid wu \in \mathrm{dom}(f)\}).$$

The *(word function) derivative of f with respect to w* is the partial function $f{\cdot}w\colon A^* \dashrightarrow B^*$ defined for all $u \in A^*$ by

$$(f{\cdot}w)(u) \quad := \quad \begin{cases} \overline{f[w]} \centerdot f(wu) & \text{if } wu \in \mathrm{dom}(f) \\ \star & \text{otherwise} \end{cases}$$

◁

4.3.28. Definition. For $f\colon A^* \dashrightarrow B^*$ define the subsequential transducer $\mathbb{T}_f = (Q, o, d, r, i, m)$ by taking:

$$\begin{array}{rclcrcl} Q & = & \{f{\cdot}w \mid w \in A^*\}, & & o(f{\cdot}w)(a) & = & \overline{f[w]} \centerdot f[wa], \\ i & = & f{\cdot}\varepsilon, & & d(f{\cdot}w)(a) & = & f{\cdot}wa, \\ m & = & f[\varepsilon], & & r(f{\cdot}w) & = & \overline{f[w]} \centerdot f(w). \end{array}$$

◁

4.3.29. Remark. In Chapter 3, we used the notation $f[a]$ and $f{\cdot}a$ to denote the initial output and derivative of a causal stream function $f\colon A^\omega \to B^\omega$ with respect to an input letter $a \in A$ (cf. Definition 3.2.1). We hope this will not cause any confusion with the usage here for maximal output and word function derivative. In principle, there is no real ambiguity due to typing (stream functions versus word functions), but more importantly, in subsection 4.4.4 (Theorem 4.4.19, Remark 4.4.20) we will see that the two notations denote the same objects up to isomorphism. ◁

In the next section, we show in Proposition 4.4.2 that $\mathsf{NSubSeq}$ is a full subcategory of $\mathsf{Coalg}(S)$ for some functor S, and that state equivalence amounts

to S-bisimilarity. It follows that for any normalised structure $\mathbb{S}$, we can obtain a minimal normalised structure by quotienting $\mathbb{S}$ with S-bisimilarity. In [37, p. 131 and 139], it was remarked[1] that minimisation of normalised subsequential transducers can be carried out by generalising existing techniques for minimising deterministic automata [66, 77], however the details were not given. We describe the minimisation of normalised structures and transducers in subsection 4.4.3.

As expected, minimal subsequential structures form a reflective subcategory of normalised subsequential structures. This result can be proved directly by showing that for any $\mathbb{S}$ in $\mathsf{NSubseq}$, the quotient of $\mathbb{S}$ with state equivalence is a minimal subsequential structure and the associated quotient map is a subsequential morphism. However, it also follows from the characterisation of $\mathsf{NSubseq}$ as a full subcategory of $\mathsf{Coalg}(S)$ together with more general results from [50], where it is proved that for any functor T, minimal T-coalgebras are reflective in $\mathsf{Coalg}(T)$. To get a reflector for $\mathsf{NSubseq}$, we just have to restrict the reflector for $\mathsf{Coalg}(S)$ to the full subcategory $\mathsf{NSubseq}$.

4.3.30. Proposition. *The full subcategory of minimal subsequential structures (transducers) is reflective in* $\mathsf{NSubseq}$ *(*$\mathsf{NSubseqTra}$*).*

Proof. The argument for minimal normalised structures was outlined in the paragraph above. The argument for the transducer case is almost identical. QED

4.4 Coalgebraisation via normalisation

One of our aims is to find out whether subsequential structures can be seen as coalgebras. A fundamental property of a category of coalgebras is that it comes with abstract definitions of morphism and state behaviour, which capture the general idea that coalgebra morphisms are behaviour preserving maps. In order to claim that a class of subsequential structures is coalgebraic, we want the notions of subsequential morphism and state behaviour (defined in Subsection 4.3.1) to coincide with the coalgebraic ones. However, based on our observations in the previous section, we already suspect that, in general, subsequential structures and morphisms are not coalgebraic, since subsequential morphisms do not preserve state behaviour, unless we find ourselves in the subcategory of normalised structures.

In this section we will first demonstrate that indeed the category $\mathsf{NSubseq}$ can be properly regarded as a category of coalgebras, whereas this does not hold for Subseq and $\mathsf{CSubseq}$. This means that the normalisation operation $\mathcal{N}$ is in

[1] Choffrut states in [37, p. 139] that a normalised subsequential transducer can be minimised by minimising the underlying automaton. This is however not true (and we assume it was just a misformulation by Choffrut), since it is clearly possible that two states are equivalent with respect to the underlying deterministic automaton while not having the same behaviour.

effect a *coalgebraisation*, and it explains why bisimilarity coincides with state equivalence, and hence that minimisation of normalised structures can be carried out by quotienting with bisimilarity. In Subsection 4.4.2 we prove that $\mathsf{NSubseq}$ has a final object. Since normalisation is a reflector it follows that this object is also final in Subseq. We also show that the existence and properties of minimal subsequential transducers (cf. Subsection 4.3.4) follow from the existence of this final object. In subsection 4.4.3 we describe how to minimise normalised structures by adapting the standard minimisation algorithm for deterministic finite automata. In Subsection 4.4.4 we show that the underlying structures of sequential transducers, and partial and total Mealy machines are coalgebras in a very precise sense, since each subclass is a category of coalgebras for some subfunctor of S. Using this observation we will show that each of these subclasses have a final object which is a substructure of the final (normalised) subsequential structure. The final objects of these subclasses are relatively well-known, and the main purpose of this subsection is to place the existing coalgebraic modelling of sequential and (partial) Mealy machines in the current context.

4.4.1 Coalgebraic modelling

Let $\mathbb{S} = (Q, o, d, r)$ be a subsequential structure. We combine o and d into a transition structure $t\colon Q \to (A \to (\mathbf{1} + B^* \times Q))$ by defining for all $q \in Q$:

$$t(q)(a) = \begin{cases} \langle o(q)(a), d(q)(a)\rangle & \text{if } a \in \mathrm{supp}(q); \\ \star & \text{otherwise} \end{cases} \tag{4.6}$$

It is then easy to see that $\mathbb{S}$ can be fully described by a single map:

$$\begin{array}{ccccccc} \langle t, r\rangle & : & Q & \to & (\mathbf{1} + B^* \times Q)^A & \times & (\mathbf{1} + B^*) \\ & & q & \mapsto & \langle \ t(q) & , & r(q) \ \rangle \end{array} \tag{4.7}$$

This map has the type of a coalgebra for the functor $S\colon \mathsf{Set} \to \mathsf{Set}$ defined by:

$$\begin{array}{lcl} S(X) & = & (\mathbf{1} + B^* \times X)^A \times (\mathbf{1} + B^*), \\ S(f\colon X \to Y) & = & (\mathbf{1} + \mathsf{Id}_{B^*} \times f)^{\mathsf{Id}_A} \times (\mathbf{1} + \mathsf{Id}_{B^*}). \end{array}$$

Clearly, every map $\langle t, r\rangle$ of the type given in (4.7) can also be seen as a subsequential structure, and from now on we will make no distinction between the two. Instantiating the definition of S-coalgebra morphism yields the following. Let $\mathbb{X}_1 = (X_1, \langle t_1, r_1\rangle)$ and $\mathbb{X}_2 = (X_2, \langle t_2, r_2\rangle)$ be S-coalgebras. A map $\alpha\colon X_1 \to X_2$ is an S-coalgebra morphism from $\mathbb{X}_1$ to $\mathbb{X}_2$ if for all $x \in X_1$:

$S(\alpha)(\langle t_1(x), r_1(x)\rangle) = \langle t_2(\alpha(x)), r_2(\alpha(x))\rangle$, which is equivalent with:

$$\begin{array}{ll}
\text{(T1)} & \forall a \in A : t_1(x)(a) = \star \iff t_2(\alpha(x))(a) = \star, \\
\text{(T2)} & \forall a \in A : t_1(x)(a) \neq \star \implies \\
 & \quad o_1(x)(a) = o_2(\alpha(x))(a) \text{ and } \alpha(d_1(x)(a)) = d_2(\alpha(x))(a), \\
\text{(R1)} & r_1(x) = \star \iff r_2(\alpha(x)) = \star, \\
\text{(R2)} & r_1(x) \neq \star \implies r_1(x) = r_2(\alpha(x)).
\end{array}$$

4.4.1. Lemma. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be two subsequential structures, and $\alpha\colon Q_1 \to Q_2$ a function. We have: α is an S-coalgebra morphism if and only if (α, ε) is a subsequential morphism.*

Proof. Straightforward, using the conditions given in Proposition 4.3.23. QED

Lemma 4.4.1 tells us that although Subseq and $\mathsf{Coalg}(S)$ have the same objects, subsequential morphisms are, in general, a strict superset of S-coalgebra morphisms, and only between normalised subsequential structures do the two coincide. In other words, only normalised subsequential structures can be properly regarded as coalgebras. Another observation which follows is that the coinduction principle for S-coalgebras is a sound proof principle for subsequential structures, but it is only complete over normalised subsequential structures.

4.4.2. Proposition. *Let $\mathbb{S}_1$ and $\mathbb{S}_2$ be subsequential structures. For all states q_1 in $\mathbb{S}_1$ and q_2 in $\mathbb{S}_2$, we have:*

1. *If $q_1 \sim_S q_2$ then $[\![q_1]\!]_1 = [\![q_2]\!]_2$.*
2. $\mathsf{NSubseq}$ *is a full subcategory of* $\mathsf{Coalg}(S)$.
3. *If $\mathbb{S}_1$ and $\mathbb{S}_2$ are normalised, then: $q_1 \sim_S q_2$ iff $[\![q_1]\!]_1 = [\![q_2]\!]_2$.*

Proof. Item (1) follows from the fact that any S-coalgebra morphism is a state behaviour preserving subsequential morphism. Item (2) follows from Proposition 4.3.23 and Lemma 4.4.1. Item (3) follows from (2) and the fact that in $\mathsf{Coalg}(S)$, behavioural equivalence coincides with S-bisimilarity. QED

Note, however, that $\mathsf{NSubseqTra}$ is not a subcategory of $\mathsf{PtCoalg}(S)$, since pointed S-coalgebras do not accomodate an initial prefix.

4.4.2 The final subsequential structure

In a category of coalgebras, a final object can be thought of as the coalgebra of behaviours. In the case of normalised subsequential structures, state behaviours

are functions $f\colon A^* \dashrightarrow B^*$ with the property that $\mathrm{lcp}(f(A^*)) = f[\varepsilon] = \varepsilon$. We call an f with this property *normalised*. We can define a normalised subsequential structure on the set of normalised functions using the notions of maximal output and derivative, cf. Definition 4.3.27.

4.4.3. Definition. We define $\Phi := \{f\colon A^* \dashrightarrow B^* \mid f[\varepsilon] = \varepsilon\}$, and $\mathbf{\Phi} = (\Phi, O, D, R)$ where

$$O(f) = f[a], \quad D(f) = f{\cdot}a, \quad R(f) = f(\varepsilon). \qquad \lhd$$

In order to verify that $\mathbf{\Phi}$ is well-defined, we check that derivatives are normalised. Let $f\colon A^* \dashrightarrow B^*$, and $a \in A$. We have

$$\begin{aligned} (f{\cdot}a)[\varepsilon] &= \mathrm{lcp}(\{\, \overline{f[a]} \centerdot f(aw) \mid aw \in \mathrm{dom}(f)\}), \text{ and} \\ f[a] &= \mathrm{lcp}(\{f(aw) \mid aw \in \mathrm{dom}(f)\}), \end{aligned}$$

hence $(f{\cdot}a)[\varepsilon] = \varepsilon$. It follows that D is well-defined, and (Φ, O, D, R) is a normalised subsequential structure.

4.4.4. Theorem. *The normalised subsequential structure $\mathbf{\Phi} = (\Phi, O, D, R)$ is a final object in the category* NSubseq *of normalised subsequential structures and subsequential morphisms, and the behaviour map $[\![\_]\!]$ is the unique subsequential morphism into $\mathbf{\Phi}$.*

Proof. Let $\mathbb{S} = (Q, o, d, r)$ be an arbitrary normalised subsequential structure, and let $[\![\_]\!] = [\![\_]\!]_{\mathbb{S}}\colon Q \to \Phi$ be the state behaviour map. We will show that $[\![\_]\!]$ is a subsequential morphism, i.e., an S-coalgebra morphism. First of all, since all states in $\mathbb{S}$ are coaccessible, $[\![\_]\!]$ is a total function. We now check that $[\![\_]\!]$ satisfies the conditions from Proposition 4.3.23. Let $q \in Q$, and $a \in A$. We have: $a \in \mathrm{supp}(q)$ iff $aw \in \mathrm{dom}([\![q]\!])$ for some $w \in A^*$ iff $\mathrm{dom}([\![q]\!]{\cdot}a) \neq \emptyset$ iff $a \in \mathrm{supp}([\![q]\!])$.

To see that $O([\![q]\!])(a) = [\![q]\!][a] = o(q)(a)$, we note that for all $w \in A^*$ such that $aw \in \mathrm{dom}([\![q]\!])$, $[\![q]\!](aw) = o(q)(a) \centerdot [\![d(q)(a)]\!](w)$, and hence

$$[\![q]\!][a] = \mathrm{lcp}([\![q]\!](aA^*)) = o(q)(a) \centerdot \mathrm{lcp}([\![d(q)(a)]\!](A^*)) = o(q)(a) \centerdot \varepsilon = o(q)(a),$$

since $d(q)(a)$ is normalised. In order to show that $[\![d(q)(a)]\!] = [\![q]\!]{\cdot}a$ for all $a \in \mathrm{supp}(q)$, let $w \in A^*$. We then have

$$\begin{aligned} [\![d(q)(a)]\!](w) &= o(d(q)(a))(w) \centerdot r(d(d(q)(a))(w)) \\ &= \overline{o(q)(a)} \centerdot o(q)(aw) \centerdot r(d(q)(aw)) \\ &= \overline{[\![q]\!][a]} \centerdot [\![q]\!](aw) \\ &= ([\![q]\!]{\cdot}a)(w). \end{aligned}$$

Finally, we have $q \in \mathrm{dom}(r)$ iff $\varepsilon \in \mathrm{dom}(\llbracket q \rrbracket)$ iff $\llbracket q \rrbracket \in \mathrm{dom}(R)$, and $R(\llbracket q \rrbracket) = \llbracket q \rrbracket(\varepsilon) = r(q)$. We leave it to the reader to verify that $\llbracket \_ \rrbracket \colon (Q, o, d, r) \to (\Phi, O, D, R)$ is unique. QED

4.4.5. Corollary. *The normalised subsequential structure* $\mathbf{\Phi} = (\Phi, O, D, R)$ *is a final object in* CSubseq *and in* Subseq.

Proof. This is an immediate consequence of Theorem 4.4.4 and the fact that NSubseq is reflective in CSubseq and Subseq, cf. Theorems 4.3.25 and 4.3.16. For $\mathbb{S} \in$ Subseq, the final map $h_{\mathbb{S}} \colon \mathbb{S} \to \mathbf{\Phi}$ is obtained by composing the reflection arrows with the behaviour map in NSubseq:

$$(h_{\mathbb{S}}, \hat{\beta}_{\mathcal{C}(\mathbb{S})}) = (\llbracket \_ \rrbracket_{\mathcal{N}(\mathcal{C}(\mathbb{S}))}, \varepsilon) \circ (id_{\mathcal{C}(\mathbb{S})}, \hat{\beta}_{\mathcal{C}(\mathbb{S})}) \circ (id_{Coacc(\mathbb{S})}, \varepsilon).$$

We recall (cf. Lemma 4.3.13) that for the reflection arrow $id_{Coacc(\mathbb{S})} \colon \mathbb{S} \to \mathcal{C}(\mathbb{S})$ the witnessing function β can be defined arbitrarily on non-coaccessible states in $\mathbb{S}$, so only if $\mathbb{S}$ is in CSubseq is the witnessing function unique and equal to $\hat{\beta}_{\mathbb{S}}$, hence there is also in general several witnessing functions for $h_{\mathbb{S}}$. Concretely, one can show that $h_{\mathbb{S}} \colon \mathbb{S} \to \mathbf{\Phi}$ is the partial map defined for all $q \in Coacc(\mathbb{S})$ by $h_{\mathbb{S}}(q) = \llbracket q \rrbracket_{\mathbb{S}} \cdot \varepsilon = \overline{\hat{\beta}_{\mathbb{S}}(q)} \centerdot \llbracket q \rrbracket_{\mathbb{S}}$. QED

4.4.6. Remark. Note that $\mathbf{\Phi}$ is not final in Coalg(S), since for an arbitrary $\mathbb{S}$ in Coalg(S), the behaviour map $\llbracket \_ \rrbracket_{\mathbb{S}}$ will not be an S-coalgebra morphism. However, since S is polynomial, we know that the final S-coalgebra exists, and it follows from Proposition 4.4.2(3) that the final Coalg(S)-morphism from $\mathbf{\Phi}$ into the final S-coalgebra is injective. Hence, as an object in Coalg(S), $\mathbf{\Phi}$ is isomorphic to a subcoalgebra of the final S-coalgebra. ◁

4.4.7. Remark. Let $\mathbb{S} = (Q, o, d, r)$ be a subsequential structure, $q \in Q$ and $w \in A^*$. Note that, in general, $\llbracket q \rrbracket_{\mathbb{S}} \cdot w \neq \llbracket d(q)(w) \rrbracket_{\mathbb{S}}$. We only have identity if $d(q)(w)$ is normalised. In particular, we have: $\llbracket q \rrbracket_{\mathbb{S}} \cdot w = \llbracket q \rrbracket_{\mathcal{N}(\mathbb{S})} \cdot w = \llbracket d(q)(w) \rrbracket_{\mathcal{N}(\mathbb{S})}$, since for all $u \in A^*$,

$$\begin{aligned}
(\llbracket q \rrbracket_{\mathbb{S}} \cdot w)(u) &= \overline{\llbracket q \rrbracket_{\mathbb{S}}(w)} \centerdot \llbracket q \rrbracket_{\mathbb{S}}(wu) \\
&= \overline{\hat{\beta}_{\mathbb{S}}(q) \centerdot \llbracket q \rrbracket_{\mathcal{N}(\mathbb{S})}(w)} \centerdot \hat{\beta}_{\mathbb{S}}(q) \centerdot \llbracket q \rrbracket_{\mathcal{N}(\mathbb{S})}(wu) \\
&= \overline{\llbracket q \rrbracket_{\mathcal{N}(\mathbb{S})}(w)} \centerdot \llbracket q \rrbracket_{\mathcal{N}(\mathbb{S})}(wu) \\
&= (\llbracket q \rrbracket_{\mathcal{N}(\mathbb{S})} \cdot w)(u).
\end{aligned}$$ ◁

We now show that the existence and properties of the minimal transducer $\mathbb{T}_f$ of a function $f \colon A^* \dashrightarrow B^*$ (cf. Definition 4.3.28) are a consequence of Theorem 4.4.4. Recall the following notation, given a function $f \colon A^* \dashrightarrow B^*$ in Φ, $\langle f \rangle_\Phi$ denotes the subcoalgebra generated by f in (Φ, O, D, R).

4.4.8. Corollary. *We have:*

1. *For any* $f\colon A^* \dashrightarrow B^*$, $\mathbb{T}_f = (\langle f{\cdot}\varepsilon\rangle_\Phi, f{\cdot}\varepsilon, f[\varepsilon])$, $\mathbb{T}_f$ *is minimal, and* $[\![\mathbb{T}_f]\!] = f$.
2. *If* $\mathbb{T} = (\mathbb{S}, i, m)$ *is an accessible subsequential transducer with* $f = [\![\mathbb{T}]\!]$, *then the final map* $h_\mathbb{S}\colon \mathbb{S} \to \mathbf{\Phi}$ *is the unique subsequential transducer morphism from* $\mathbb{T}$ *onto the minimal subsequential transducer* $\mathbb{T}_f$.
3. *Two accessible subsequential transducers* $\mathbb{T}_1$ *and* $\mathbb{T}_2$ *are equivalent if and only if there exists a subsequential transducer* $\mathbb{T}$ *and subsequential morphisms* $\alpha_j\colon \mathbb{T}_j \to \mathbb{T}$, *for* $j \in \{1, 2\}$.

Proof. *Item (1):* The proof that $\mathbb{T}_f = (\langle f{\cdot}\varepsilon\rangle_\Phi, f{\cdot}\varepsilon, f[\varepsilon])$ is almost immediate from Definitions 4.3.28 and 4.4.3. For example, to check that the output functions in Definitions 4.3.28 and 4.4.3 are the same, let $f{\cdot}w \in \langle f{\cdot}\varepsilon\rangle_\Phi$. We have for any $a \in A$:

$$O(f{\cdot}w)(a) = (f{\cdot}w)[a] = \mathrm{lcp}(\{\overline{f[w]} \centerdot f(wau) \mid u \in A^*\}) = \overline{f[w]} \centerdot f[wa].$$

Minimality of $\langle f{\cdot}\varepsilon\rangle_\Phi$ follows from the finality of $\mathbf{\Phi}$. Finally, for all $w \in A^*$, $w \in \mathrm{dom}(f)$ iff $w \in \mathrm{dom}(f{\cdot}\varepsilon) = \mathrm{dom}(\mathbb{T}_f)$, and for $w \in \mathrm{dom}(f)$ we have:

$$[\![\mathbb{T}_f]\!](w) = f[\varepsilon] \centerdot [\![f{\cdot}\varepsilon]\!](w) = f[\varepsilon] \centerdot (f{\cdot}\varepsilon)(w) = f[\varepsilon] \centerdot \overline{f[\varepsilon]} \centerdot f(w) = f(w).$$

Item (2): The proof for trimmed $\mathbb{T}$ can be found in [37]. The result for accessible $\mathbb{T}$ follows from the finality of $\mathbf{\Phi}$. The final map $h_\mathbb{S}$ is a unique subsequential morphism from $\mathbb{S}$ to $\langle f{\cdot}\varepsilon\rangle_\Phi$ with a witnessing function β such that $\beta{\restriction}_{Coacc(\mathbb{S})} = \hat{\beta}_{\mathcal{C}(\mathbb{S})}$. In the case the initial state of $\mathbb{T}$ is not coaccessible, $\mathbb{T}_f$ is the empty transducer, and the final map $h_\mathbb{S}$ is the empty map. Otherwise, it follows that $(h_\mathbb{S}, \beta)$ also satisfy (ε-in) since $f[\varepsilon] = m \centerdot \hat{\beta}_\mathbb{S}(i)$, and (init) since:

$$h_\mathbb{S}(i) = \overline{\hat{\beta}_\mathbb{S}(i)} \centerdot [\![i]\!]_\mathbb{S} = \overline{\hat{\beta}_\mathbb{S}(i)} \centerdot \overline{m} \centerdot f = \overline{m \centerdot \hat{\beta}_\mathbb{S}(i)} \centerdot f = \overline{f[\varepsilon]} \centerdot f = f{\cdot}\varepsilon.$$

Item (3): The direction from right to left follows from the fact that subsequential transducer morphisms preserve behaviour (Proposition 4.3.9); the other direction follows from item (2). QED

4.4.3 Minimisation algorithm for normalised structures

As we remarked in subsection 4.3.4, normalised subsequential structures can be minimised by quotienting with state equivalence, and the reason is that in a normalised $\mathbb{S}$, state equivalence coincides with the largest congruence on $\mathbb{S}$, which in turn coincides with S-bisimilarity (cf. Proposition 4.4.2). We now

describe how we can compute the state equivalence relation on an S-coalgebra (and hence on a normalised subsequential structure) by adapting the existing method for computing state equivalence on deterministic finite automata (DFA).

First of all, working out the details of the definition of S-bisimulation yields the following. Given an S-coalgebra $\mathbb{S} = (Q, o, d, r)$, S-bisimilarity on $\mathbb{S}$ is the largest relation $R \subseteq Q \times Q$ such that for any two states $q_1, q_2 \in Q$, we have: $\langle q_1, q_2 \rangle \in R$ implies

(s0) $\mathrm{supp}(q_1) = \mathrm{supp}(q_2)$;

(s1) for all $a \in \mathrm{supp}(q_1) : o(q_1)(a) = o(q_2)(a)$;

(s2) $r(q_1) = r(q_2)$; and

(s3) for all $a \in \mathrm{supp}(q_1) : \langle d(q_1)(a), d(q_2)(a) \rangle \in R$.

In order to make the connection with the algorithm for DFA's clear, we recall the definition of bisimilarity for deterministic automata (DA-bisimilarity), i.e., bisimilarity for the functor $2 \times (-)^A$ (cf. Subsection 2.4.2). Let $\mathbb{A} = (Q, d, F)$ be a deterministic automaton. DA-bisimilarity on $\mathbb{A}$ is the largest relation $R \subseteq Q \times Q$ such that for any two states $q_1, q_2 \in Q$, $\langle q_1, q_2 \rangle \in R$ implies that:

(a1) $o(q_1) = o(q_2)$ (i.e. $q_1 \in F$ iff $q_2 \in F$); and

(a2) for all $a \in A : \langle d(q_1)(a), d(q_2)(a) \rangle \in R$.

In [126] it was shown that in a DA, state equivalence is DA-bisimilarity. Computing the state equivalence relation on a DFA has been presented in several places and variations (see e.g. [66, 77] for textbook presentations). The main idea is the following. The computation starts with the partition $P_0 = \{F, Q \setminus F\}$ of Q, where $Q \setminus F$ is the complement of F in Q. The subsets in P_0 are the equivalence classes of the largest equivalence relation such that condition (a1) holds for all P_0-related states. In the main loop, P_0 is iteratively refined into an equivalence relation which also satisfies condition (a2). This is done by inspecting the current partition $P_k = \{Q_1, \ldots, Q_n\}$ of Q for the existence of some $Q_i, Q_j \in P_k$ and $a \in A$ such that there are $q, s \in Q_i$ for which $d(q)(a) \in Q_j$ and $d(s)(a) \notin Q_j$. In that case, Q_i is split (by (Q_j, a)) into the two sets $Q_i' = \{q \in Q_i \mid d(q)(a) \in Q_j\}$ and $Q_i'' = \{q \in Q_i \mid d(q)(a) \notin Q_j\}$, in other words, P_k is refined into $P_{k+1} = (P_k \setminus \{Q_i\}) \cup \{Q_i', Q_i''\}$. This refinement process continues until no more splits can be made. When this happens, the partition stores the A-bisimilarity classes on $\mathbb{A}$. By using extra datastructures it is possible to choose the *splitters* (Q_j, a) wisely, and reduce the number of actual splits that must be carried out, resulting in an algorithm which runs in time $O(|A|\, n \log(n))$ where n is the number of states, and $|A|$ is the size of the input alphabet A (cf. [74], see also [65, 48]).

The adaptation of the DFA-algorithm to S-coalgebras consists in changing the initial partition. The refinement part of the algorithm stays the same. We take as initial partition the (classes of) the largest equivalence relation P_0^s on Q such that all pairs related by P_0^s satisfy (s0), (s1) and (s2). Running the refinement algorithm starting from this initial partition will result in the largest equivalence relation which also satisfies (s3), i.e., the S-bisimilarity relation on $\mathbb{S}$. This can be proved by essentially the same argument used for the correctness of the algorithm for DFA's, see e.g. [74, proof of Prop. 5]. Moreover, P_0^s is clearly a refinement of the P_0-partition associated with the underlying DA, hence the refinement of P_0^s into S-bisimilarity will also terminate in time $O(|A|\, n \log(n))$. It remains to describe how to compute P_0^s.

4.4.9. Lemma. *Given a finite S-coalgebra $\mathbb{S} = (Q, o, d, r)$, we can compute the largest equivalence relation P_0^s on Q which satisfies (s0), (s1) and (s3) in time $O((|A|\|o\| + \|r\|)|Q| \log(|Q|))$, where $\|o\| := \max\{|o(q)(a)| \mid q \in Q, a \in A\}$ and $\|r\| := \max\{|r(q)| \mid q \in Q\}$.*

Proof. We want to group together states which have the same output function and the same terminal output. This can be done efficiently by a variation on a sorting algorithm which can be implemented using a balanced binary search tree (cf. [73]). The nodes of a binary search tree T are pairs (c, l_c) where c is a key and l_c is a value, and it is required that a linear order $<^T$ exists on the set of all keys. The tree operations are implemented to maintain an ordering based on key-values, such that for a node (c, l_c), all key values in the left subtree are less than c and all key values in the right subtree are greater than c. This means that looking up a key value can be done in time proportional with the height of the tree. In a balanced binary search tree with n nodes, the height is at most $\log(n)$.

Since $\mathbb{S}$ is finite, we can assume $A = \{a_1, a_2, \ldots, a_k\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ by enumerating all input and output letters that occur in $\mathbb{S}$. In our case, a key c is a data record which stores the output and terminal output functions for some state q. We define $c(q) := \langle o(q)(a_1), \ldots, o(q)(a_k), r(q)\rangle$. The value l_c associated with a key c will be a list of states q such that $c(q) = c$. We will use a variation on insertion which does the following. Inserting $(c(q), q)$ into a tree which contains a node $(c', l_{c'})$ with $c' = c(q)$ will result in adding q to l_c, and if $c(q)$ does not occur in the tree, then a new node $(c(q), \{q\})$ is added. Once we can define a linear ordering on all $c(q)$-values, P_0^s can be computed by inserting $(c(q), q)$ for all states q into an initially empty tree. In the resulting tree T at each node (c, l_c) the list l_c contains the elements of the P_0^s-equivalence class corresponding with the value c, and we can obtain P_0^s by traversing T and retrieving the node values l_c.

It remains to define a linear ordering on the key values, which are elements of $(\mathbf{1} + B^*)^{k+1}$. We define a linear order $\prec$ on A and B by $a_1 \prec a_2 \prec \ldots \prec a_k$, and

$b_1 \prec b_2 \prec \ldots \prec b_n$. We extend this ordering to the lexicographic ordering on B^*, which we also denote $\prec$. In order to deal with the partiality of the transition structure, we extend $\prec$ to $\mathbf{1} + B^*$ by defining $w \prec \star$ for all $w \in B^*$. Finally, we can lift $\prec$ to the corresponding lexicographic ordering on $(\mathbf{1} + B^*)^{k+1}$. For example, if $a \prec b \prec \star$, then $\langle a, b, \star\rangle \prec \langle a, \star, b\rangle \prec \langle aa, \star, b\rangle$.

We now analyse the complexity of computing P_0^s. An insertion in a balanced binary search tree with n nodes can be done in time $O(C \log(n))$, where C is an upper bound for the cost of comparing keys. In our case, a key has the form $c(q) = \langle o(q)(a_1), \ldots, o(q)(a_k), r(q)\rangle$ and hence its size is bounded by $|A|\|o\|+\|r\|$. The highest comparison cost is incurred when $c(q)$ is already present in the tree, since then each component in $c(q)$ will be successfully matched till the end. Each comparison can thus be done in time $O(|A|\|o\| + \|r\|))$. We have $|Q|$ elements to insert, hence the tree T can be constructed in time $O((|A|\|o\|+\|r\|)|Q| \log(|Q|))$. QED

It is difficult to give a compact description of the overall time complexity of carrying out minimisation via normalisation due to the many different factors involved. So in the next proposition we just provide the upper bounds for the two main components of the algorithm. In particular, we do not include the cost of actually constructing the quotient structure.

4.4.10. Proposition. *Let $\mathbb{S} = (Q, o, d, r)$ be a finite subsequential structure. Normalising $\mathbb{S}$ and computing state equivalence on $\mathbb{S}$ can be done in time:*

Compute $\mathcal{N}(\mathbb{S})$:	$O((\|\hat{\beta}\| + 1)M)$
Compute state equivalence on $\mathcal{N}(\mathbb{S})$:	$O((\|A\|\,\|o\| + \|r\|)\|Q\| \log(\|Q\|))$

where $\|\hat{\beta}\| = \max\{|\hat{\beta}(q)| \mid q \in Q\}$ and M is the number of transitions in $\mathbb{S}$.

Proof. For the normalisation part, we refer to the time complexity of the algorithm given by Béal & Carton [17], see also [37]. To compute the state equivalence relation on $\mathcal{N}(\mathbb{S})$, we need $O((|A|\,\|o\| + \|r\|)|Q| \log(|Q|))$ time for computing P_0^s (Lemma 4.4.9), and $O(|A||Q| \log(|Q|))$ time for completing the refinement stage. Since $O(|A||Q| \log(|Q|))$ is dominated by $O((|A|\,\|o\| + \|r\|)|Q| \log(|Q|))$ this gives us the claimed upper bound for computing state equivalence. QED

4.4.4 Sequential transducers and Mealy machines

Sequential transducers can be identified with the subclass of subsequential transducers that have no initial and terminal output, nor internal states. Eilenberg [41] gives a detailed treatment of sequential transducers under the name *generalised sequential machines*. In particular, in Chapter XII of [41] Eilenberg proves the existence of a final sequential structure, although he never uses the

words finality or coalgebra. Yet another familiar subclass consists of (partial) Mealy machines, which are sequential transducers that always output a single letter (rather than a word) on each transition. The coalgebraic modelling of Mealy machines with a total transition structure was presented in Chapter 3 (see also [132]). In this subsection we will see that sequential structures, partial Mealy structures and total Mealy structures can be characterised as coalgebras for subfunctors of S. Once we know this, it follows that each of these subclasses contain a final object which is a substructure of the final normalised structure $\mathbf{\Phi}$.

4.4.11. Definition. A subsequential structure $\mathbb{S} = (Q, o, d, r)$ is

– *sequential* if $\mathrm{dom}(r) = Q$ and for all $q \in Q$: $r(q) = \varepsilon$.
– *partial Mealy* if $\mathbb{S}$ is sequential and for all $q \in Q$ and $a \in A$: $o(q)(a) \in B$.
– *Mealy* if $\mathbb{S}$ is partial Mealy and for all $q \in Q$: $\mathrm{supp}(q) = A$.

We denote the corresponding full subcategories of Subseq by Seq, PMealy and Mealy, respectively. A *sequential (partial Mealy, Mealy)* transducer is a subsequential transducer $\mathbb{T} = (\mathbb{S}, i, m)$ where $\mathbb{S}$ is sequential (partial Mealy, Mealy) and $m = \varepsilon$. ◁

4.4.12. Remark. Usually, (partial) Mealy transducers are referred to as (partial) Mealy machines, and we will also use this terminology when convenient. In the present context, we do not require Mealy machines to be finite. The initial prefix m and the terminal output function r are trivially defined for sequential transducers, we will therefore sometimes leave them out of the description of a sequential transducer, and simply write $\mathbb{T} = (Q, o, d, i)$. Similarly, a sequential structure will be denoted $\mathbb{S} = (Q, o, d)$. ◁

From Definition 4.4.11 it is immediate that a sequential structure $\mathbb{S}$ is normalised, since for all states q in $\mathbb{S}$, $\hat{\beta}(q) = r(q) = \varepsilon$. Hence $\mathsf{Mealy} \sqsubseteq \mathsf{PMealy} \sqsubseteq \mathsf{Seq}$ form a sequence of nested full subcategories of $\mathsf{NSubseq}$, and their morphisms are characterised by Proposition 4.3.23. For simplicity, we will refer to these morphisms as *Mealy, partial Mealy and sequential morphisms*, respectively.

The coalgebraic modelling of sequential, partial Mealy and Mealy structures is straightforward. Consider the following subfunctors of S:

$$\begin{aligned} S_0(X) &= (\mathbf{1} + B^* \times X)^A \times \{\varepsilon\}, \\ M_0(X) &= (\mathbf{1} + B \times X)^A \times \{\varepsilon\}, \\ M(X) &= (B \times X)^A \times \{\varepsilon\}. \end{aligned}$$

Using the embedding of B into B^* and the canonical embedding of a set X into $\mathbf{1} + X$ it is also clear that $M \hookrightarrow M_0 \hookrightarrow S_0$, and hence $\mathsf{Coalg}(M) \sqsubseteq \mathsf{Coalg}(M_0) \sqsubseteq \mathsf{Coalg}(S_0)$.

4.4.13. Proposition. *We have:*

1. $\mathsf{Seq} \cong \mathsf{Coalg}(S_0)$,
2. $\mathsf{PMealy} \cong \mathsf{Coalg}(M_0)$,
3. $\mathsf{Mealy} \cong \mathsf{Coalg}(M)$.

Proof. It is easily verified that given a sequential structure $\mathbb{S} = (Q, o, d, r)$ the transition structure $\langle t, r\rangle$, as defined in (4.6), is a map of the type $\langle t, r\rangle \colon Q \to S_0(Q)$, that is, $\mathbb{S}$ is an object in $\mathsf{Coalg}(S_0)$. Hence Seq and $\mathsf{Coalg}(S_0)$ have the same objects. It now follows from $\mathsf{Seq} \sqsubseteq \mathsf{NSubseq} \sqsubseteq \mathsf{Coalg}(S)$ and $\mathsf{Coalg}(S_0) \sqsubseteq \mathsf{Coalg}(S)$ that a sequential morphism is a $\mathsf{Coalg}(S_0)$-morphism and vice versa. The items for PMealy and Mealy follow with a similar argument. QED

Hence, unlike normalised subsequential structures, Seq, PMealy and Mealy are not just full subcategories of $\mathsf{Coalg}(S)$, but themselves a category of coalgebras for a (polynomial) functor. The subobject relationship extends to the final objects of Seq, PMealy and Mealy.

4.4.14. Lemma. *Let $F, G \colon \mathsf{Set} \to \mathsf{Set}$ be polynomial functors. If $F \hookrightarrow G$ then the final F-coalgebra is isomorphic to a subcoalgebra of the final G-coalgebra.*

Proof. Let $\mathbf{\Phi}_F$ and $\mathbf{\Phi}_G$ denote the final F-coalgebra and the final G-coalgebra, respectively. Recall that these exist since F and G are polynomial. It suffices to show that the final G-coalgebra morphism from $\mathbf{\Phi}_F$ to $\mathbf{\Phi}_G$ is injective. In [16, Theorem 9], it was shown that if $F \hookrightarrow G$ and F preserves weak pullbacks, then for any F-coalgebra (X, ξ) and states $x, y \in X$, $x \sim_G y$ implies $x \sim_F y$. Since all polynomial functors preserve weak pullbacks, it follows that for all s, t in $\mathbf{\Phi}_F$, if $s \sim_G t$ then $s \sim_F t$ and by finality of $\mathbf{\Phi}_F$, $s = t$. We have thus shown that the final G-coalgebra morphism from $\mathbf{\Phi}_F$ to $\mathbf{\Phi}_G$ is injective, and the result follows. QED

The above lemma does not tell us directly that the final objects of Seq, PMealy and Mealy are subobjects of the final subsequential structure $\mathbf{\Phi}$, since $\mathbf{\Phi}$ is not the final S-coalgebra, however, a similar argument shows that this is the case. If X and Y are objects in some subcategory C, we will write $X \hookrightarrow Y$ if X is a subobject of Y in C.

4.4.15. Proposition. *Let $\mathbf{\Phi}_{\mathsf{Seq}}$, $\mathbf{\Phi}_{\mathsf{PMealy}}$, $\mathbf{\Phi}_{\mathsf{Mealy}}$ and $\mathbf{\Phi}_S$ denote final objects of Seq, PMealy, Mealy and $\mathsf{Coalg}(S)$, respectively. We have in $\mathsf{Coalg}(S)$:*

$$\mathbf{\Phi}_{\mathsf{Mealy}} \hookrightarrow \mathbf{\Phi}_{\mathsf{PMealy}} \hookrightarrow \mathbf{\Phi}_{\mathsf{Seq}} \hookrightarrow \mathbf{\Phi} \hookrightarrow \mathbf{\Phi}_S.$$

PROOF. From Lemma 4.4.14 we get: $\mathbf{\Phi}_{\mathsf{Mealy}} \hookrightarrow \mathbf{\Phi}_{\mathsf{PMealy}} \hookrightarrow \mathbf{\Phi}_{\mathsf{Seq}} \hookrightarrow \mathbf{\Phi}_S$. In Remark 4.4.6 it was shown that $\mathbf{\Phi} \hookrightarrow \mathbf{\Phi}_S$. Finally, $\mathbf{\Phi}_{\mathsf{Seq}} \hookrightarrow \mathbf{\Phi}$ holds since the behaviour map from $\mathbf{\Phi}_{\mathsf{Seq}}$ to $\mathbf{\Phi}$ is an S-coalgebra morphism, and by Proposition 4.4.2(3) and $\mathbf{\Phi}_{\mathsf{Seq}} \hookrightarrow \mathbf{\Phi}_S$, it must be injective. QED

4.4.16. REMARK. We note that since $\mathbf{\Phi}_{\mathsf{Seq}}$, $\mathbf{\Phi}_{\mathsf{PMealy}}$ and $\mathbf{\Phi}_{\mathsf{Mealy}}$ are also objects in $\mathsf{NSubseq}$, and $\mathsf{NSubseq}$ is full in Subseq, the first three substructure relationships of Proposition 4.4.15 also hold in Subseq. ◁

We will now give concrete representations of the final objects of Seq, PMealy and Mealy by characterising the behaviours of each subclass. First we show a useful property of prefix-preserving functions.

4.4.17. LEMMA. *Let $f\colon A^* \dashrightarrow B^*$ be a prefix-preserving function. For all $w \in \mathrm{dom}(f)$, $f[w] = f(w)$.*

PROOF. From the definition of $f[w]$ it is clear that $f[w] \preceq f(w)$. Since f is prefix-preserving, we have for all $u \in A^*$, such that $wu \in \mathrm{dom}(f)$, that $f(w) \preceq f(wu)$. Hence also $f(w) \preceq f[w]$, and so $f(w) = f[w]$. QED

4.4.18. PROPOSITION. *Let $f\colon A^* \dashrightarrow B^*$ be a partial function. We have:*

1. *f is computed by a sequential transducer if and only if f is prefix-preserving and $f(\varepsilon) = \varepsilon$.*
2. *f is computed by a partial Mealy machine if and only if f is length- and prefix-preserving.*
3. *f is computed by a Mealy machine if and only if f is total, and length- and prefix-preserving.*

PROOF. Item (1): Let $\mathbb{T} = (Q, o, d, i)$ be a sequential transducer and $f = [\![\mathbb{T}]\!]$. Since all states in $\mathbb{T}$ are final, the domain of f is clearly prefix-closed. Now let $uw \in \mathrm{dom}(f)$ for $u, w \in A^*$. By Definition 4.3.2, $f(uw) = o(i)(uw) = o(i)(u) \cdot o(d(i)(u))(w) = f(u) \cdot o(d(i)(u))(w)$. Hence f is prefix-preserving, and clearly $f(\varepsilon) = \varepsilon$. To show the converse, assume f is prefix-preserving and $f(\varepsilon) = \varepsilon$. We will show that the minimal transducer $\mathbb{T}_f$ is sequential, which amounts to showing that $f[\varepsilon] = \varepsilon$ and for all $w \in \mathrm{dom}(f)$, $\overline{f[w]} \cdot f(w) = \varepsilon$. Since $f(\varepsilon) = \varepsilon$, both follow from Lemma 4.4.17.

Item (2): The direction from left to right is easily verified using the definition of behaviour. For the converse, assume f is length- and prefix-preserving. We now show that the minimal transducer $\mathbb{T}_f$ is partial Mealy. From item (1) we know that $\mathbb{T}_f$ is sequential. It remains to show that for all $wa \in \mathrm{dom}(f)$, $o(f{\cdot}w)(a) = \overline{f[w]} \cdot f[wa] \in B$. From Lemma 4.4.17, we get that for

all $w \in \mathrm{dom}(f)$, $\overline{f[w]} \cdot f[wa] = \overline{f(w)} \cdot f(wa)$. Now, since f is length- and prefix-preserving it follows that $\overline{f(w)} \cdot f(wa) \in B$.

Item (3): Again, we leave the easy direction from left to right to the reader. The converse direction follows from item (2) and the observation that in $\mathbb{T}_f$, for all $w \in A^*$ and $a \in A$, $a \in \mathrm{supp}(f \cdot w)$ if and only if $wa \in \mathrm{dom}(f)$, which is always the case when f is total. Hence $\mathbb{T}_f$ is a Mealy machine. QED

Recall (from the Preliminaries, subsection 2.2) that subcoalgebras are determined by their carrier set. For the polynomial functor S it is straightforward to confirm that subcoalgebras are the transition closed subsets, and if (Ψ, ψ) is a subcoalgebra of $\mathbf{\Phi}$, then $(\Psi, \psi) = \mathbf{\Phi}{\restriction_\Psi}$, the restriction of $\mathbf{\Phi}$ to Ψ.

4.4.19. Theorem. *We have:*

1. *Let* $\Phi_{\mathsf{Seq}} = \{f \colon A^* \dashrightarrow B^* \mid f$ *is prefix-preserving and* $f(\varepsilon) = \varepsilon\}$. Φ_{Seq} *is a subcoalgebra of* $\mathbf{\Phi}$ *and a final object in* Seq.
2. *Let* $\Phi_{\mathsf{PMealy}} = \{f \colon A^* \dashrightarrow B^* \mid f$ *is length- and prefix-preserving*$\}$. Φ_{PMealy} *is a subcoalgebra of* $\mathbf{\Phi}$ *and a final object in* PMealy.
3. *Let* $\Phi_{\mathsf{Mealy}} = \{f \colon A^* \to B^* \mid f$ *is length- and prefix-preserving*$\}$. Φ_{Mealy} *is a subcoalgebra of* $\mathbf{\Phi}$ *and a final object in* Mealy.

Proof. Item (1): First, $\Phi_{\mathsf{Seq}} \subseteq \Phi$ since $f(\varepsilon) = \varepsilon$ implies that $f[\varepsilon] = \varepsilon$. If Φ_{Seq} is a subcoalgebra of $\mathbf{\Phi}$, then Φ_{Seq} is a sequential structure, since for all $f \in \Phi_{\mathsf{Seq}}$, $R(f) = f(\varepsilon) = \varepsilon$. To see that Φ_{Seq} is transition closed, let $f \in \Phi_{\mathsf{Seq}}$. From Proposition 4.4.18(1) it follows that f and any derivative of f is computed by a sequential transducer, hence any derivative of f is in Φ_{Seq}. Also from Proposition 4.4.18(1) we know that for any sequential structure $\mathbb{S}$, the behaviour map $[\![\_]\!] \colon \mathbb{S} \to \mathbf{\Phi}$ factors via Φ_{Seq}, i.e., $[\![\_]\!] \colon \mathbb{S} \to \mathbf{\Phi}_{\mathsf{Seq}}$ is a sequential morphism, and it must be unique due to the finality of $\mathbf{\Phi}$. Items (2) and (3) follow with similar arguments. QED

4.4.20. Remark. In Chapter 3, subsection 3.2.2, we saw that a final Mealy coalgebra $\mathbf{\Gamma} = (\Gamma, \gamma)$ is obtained by equipping the set $\Gamma = \{f \colon A^\omega \to B^\omega \mid f$ causal $\}$ with the operations initial output and stream function derivative (cf. Theorem 3.2.2, page 25). Since final objects are unique up to isomorphism, it follows that in Mealy, $\mathbf{\Phi}{\restriction_{\Phi_{\mathsf{Mealy}}}}$ is isomorphic to $\mathbf{\Gamma}$. This isomorphism can be made explicit by writing out the details of the behaviour map $[\![\_]\!]_{\mathbf{\Gamma}} \colon \mathbf{\Gamma} \to \mathbf{\Phi}{\restriction_{\Phi_{\mathsf{Mealy}}}}$, and showing that it is a bijection. We leave the details to the reader. ◁

4.4.21. Remark. The word function semantics of Φ_{Mealy} generalises neatly to partial Mealy structures, as we have seen in Theorem 4.4.19, but it is not the case that the set $\mathcal{X} = \{f \colon A^\omega \dashrightarrow B^\omega\}$ is a final partial Mealy structure with initial

output and derivative. In particular, the behaviour map from $\mathcal{X}$ to Φ_{PMealy} is not surjective. To see this, observe that for every $f \in \mathcal{X}$, if $\alpha \in \mathrm{dom}(f)$, then all finite prefixes of α are in $\mathrm{dom}(\llbracket f \rrbracket)$ It follows that for $g \in \Phi_{\mathsf{PMealy}}$ with $\mathrm{dom}(g)$ finite and non-empty, there is no $f \in \mathcal{X}$ such that $\llbracket f \rrbracket = g$, and such $g \in \Phi_{\mathsf{PMealy}}$ clearly exist. ◁

4.5 Coalgebraisation via differentials

The reason why subsequential structures, in general, cannot be seen as coalgebras is essentially due to the fact that their semantics allows for asynchrony at internal computation steps, whereas the coalgebraic notion of equivalence requires synchrony at all steps. We have seen that normalisation is one way of eliminating internal asynchrony. In this section, we will see that there is an alternative coalgebraic representation of the class of subsequential structures which have no internal states, and therefore also no proper internal computations. We call this subclass *step-by-step structures*. The coalgebraic representation is obtained by generalising the differential of sequential functions (cf. [41]) to subsequential functions with prefix-closed domain. This generalisation of the differential was introduced in [122] and has since been used to characterise subsequential functions in [29]. We will show that the differential representation can be used to determine equivalence of step-by-step transducers without having to normalise (Theorem 4.5.22). The main advantage of using this alternative method is that in contrast with normalisation, which requires a global fixed point computation of $\hat{\beta}$ (see subsection 4.3.3), the automaton transformation which corresponds with taking differentials can be computed locally. Finally, we will show that the minimal differential representation of a step-by-step $\mathbb{S}$ is the differential representation of the minimisation of $\mathbb{S}$.

4.5.1 Step-by-step structures

4.5.1. Definition. A subsequential structure $\mathbb{S} = (Q, o, d, r)$ is called *step-by-step* if $\mathrm{dom}(r) = Q$ (i.e. all states are final). A subsequential transducer $(\mathbb{S}, i, m)$ is step-by-step if $\mathbb{S}$ is step-by-step. ◁

4.5.2. Example. Consider the following two simple step-by-step subsequential transducers.

The behaviour of both $\mathbb{T}_3$ and $\mathbb{T}_4$ is the partial function $f\colon \{a,b\}^* \dashrightarrow \{a,b\}^*$ with $\mathrm{dom}(f) = \{b^k, b^k a \mid k \in \omega\}$ where $f(b^k) = ba^{k+1}b$ and $f(b^k a) = ba^{k+2}b$ for $k \in \omega$. ◁

The above example shows that step-by-step subsequential transducers are not necessarily normalised, hence two step-by-step subsequential transducers can computee the same function without being in perfect synchrony. Nevertheless, their morphisms can be characterised without the explicit reference to an output shift function β.

4.5.3. Proposition. *Let $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1)$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ be step-by-step subsequential structures. A function $\alpha\colon Q_1 \to Q_2$ is a subsequential morphism if and only if for all $q \in Q_1$ the following hold:*

$$
\begin{array}{ll}
\text{(supp)} & \mathrm{supp}(q) = \mathrm{supp}(\alpha(q)), \\
(\text{next})_{\mathsf{C}} & \forall a \in \mathrm{supp}(q)\colon \alpha(d_1(q)(a)) = d_2(\alpha(q))(a), \\
(\text{out})_{\mathsf{S}} & \forall a \in \mathrm{supp}(q)\colon \overline{r_1(q)} \centerdot o_1(q)(a) \centerdot r_1(d_1(q)(a)) = \\
 & \qquad\qquad \overline{r_2(\alpha(q))} \centerdot o_2(\alpha(q))(a) \centerdot r_2(d_2(\alpha(q))(a)), \\
(\text{term-out})_{\mathsf{S}} & r_1(q) \centerdot \overline{r_2(\alpha(q))} \in B^*.
\end{array}
$$

where equality in $(\text{out})_{\mathsf{S}}$ is in the free group $B^{(*)}$. Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ be step-by-step subsequential transducers. A function $\alpha\colon Q_1 \to Q_2$ is a subsequential (transducer) morphism from $\mathbb{T}_1$ to $\mathbb{T}_2$ if and only if $\alpha\colon \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism and whenever $\mathbb{T}_1$ and $\mathbb{T}_2$ are not empty:

$$
\begin{array}{ll}
\text{(init)} & \alpha(i_1) = i_2, \\
(\varepsilon\text{-in})_{\mathsf{S}} & m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2).
\end{array}
$$

Proof. First assume $\alpha\colon \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism with witnessing function $\beta\colon Q_1 \to B^*$. Note that since step-by-step structures are coaccessible, α satisfies the conditions from Proposition 4.3.14, in particular, α must be a total function. Condition (term-out) implies that for all $q \in Q_1$:

$$\beta(q) = r_1(q) \centerdot \overline{r_2(\alpha(q))}. \tag{4.8}$$

Hence, as $\beta(q) \in B^*$, $(\text{term-out})_{\mathsf{S}}$ must hold. Using (4.8), one can also easily verify that $(\text{out})_{\mathsf{C}}$ reduces to $(\text{out})_{\mathsf{S}}$.

Conversely, for any total function $\alpha\colon Q_1 \to Q_2$ which satisfies the above requirements, we can define $\beta\colon Q \to B^*$ using (4.8), since condition $(\text{term-out})_{\mathsf{S}}$ guarantees that $\beta(q) \in B^*$. It is now straightforward to verify that $(\alpha, \beta)\colon \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism.

Finally, a subsequential transducer morphism $\alpha\colon \mathbb{T}_1 \to \mathbb{T}_2$ between step-by-step transducers satisfies $(\varepsilon\text{-in})_{\mathsf{S}}$ due to (4.8) and (ε-in). Conversely, if $\alpha\colon \mathbb{S}_1 \to$

$\mathbb{S}_2$ is a subsequential morphism satisfying $(\varepsilon\text{-in})_{\mathsf{S}}$, then $(\varepsilon\text{-in})$ must hold for the unique witnessing function β given in (4.8). QED

Let Step denote the full subcategory of CSubseq which has step-by-step subsequential structures as its objects, similarly, StepTra denotes the full subcategory of CSubseqTra which has step-by-step subsequential transducers as its objects.

The behaviours of step-by-step transducers do not preserve prefixes, as illustrated in Example 4.5.2. However, since all states are final, the domain is prefix-closed.

4.5.4. Proposition. *A function $f\colon A^* \dashrightarrow B^*$ is computed by a step-by-step subsequential transducer if and only if* $\mathrm{dom}(f)$ *is prefix-closed.*

Proof. Clearly, if f is computed by a step-by-step subsequential transducer, then $\mathrm{dom}(f)$ is prefix-closed. To prove the other direction, it suffices to show that the minimal subsequential transducer $\mathbb{T}_f$ is step-by-step, that is, for all $w \in A^*$, if $w \in \mathrm{dom}(f)$ then $\varepsilon \in \mathrm{dom}(f{\cdot}w)$. But this is immediate from the definition of $f{\cdot}w$, cf. Definition 4.3.27 (page 87). QED

4.5.2 Differential representations

Although step-by-step behaviours do not preserve prefixes, they have a property which generalises the following basic decomposition property of prefix-preserving functions. If $f\colon A^* \dashrightarrow B^*$ is prefix-preserving then for all $w = a_1a_2\ldots a_n \in A^*$, $f(w)$ factors as:

$$f(w) = f(\varepsilon)\centerdot f(a_1)\centerdot (f{\cdot}a_1)(a_2)\centerdot (f{\cdot}a_1a_2)(a_3)\centerdot \ \ldots\ \centerdot (f{\cdot}a_1a_2\ldots a_{n-1})(a_n). \tag{4.9}$$

The differential of a prefix-preserving f describes the growth of f and is formally defined as the map $D_f\colon A^+ \dashrightarrow B^*$ which for all $wa \in \mathrm{dom}(f)$ is determined by the equation $f(wa) = f(w)\centerdot D_f(wa)$ (cf. [41]), that is, $D_f(wa) = \overline{f(w)}\centerdot f(wa)$. Recall from Lemma 4.4.17 that for a prefix-preserving f, $f[w] = f(w)$, and hence $D_f(wa) = (f{\cdot}w)(a)$. We can now rewrite (4.9) as:

$$f(w) = f(\varepsilon)\centerdot D_f(a_1)\centerdot D_f(a_1a_2)\centerdot \ \ldots\ \centerdot D_f(a_1a_2\ldots a_n). \tag{4.10}$$

If f does not preserve prefixes, we may not be able to decompose f-values as in (4.9). For example, if $f(a_1)$ is not a prefix of $f(a_1a_2)$, then $f(a_1a_2) \neq f(a_1)\centerdot (f{\cdot}a_1)(a_2)$. However, if f has a prefix-closed domain, f can still be decomposed using the differential by allowing D_f to take values in the free group $B^{(*)}$ rather than B^*.

4.5.5. Definition. Let $f\colon A^* \dashrightarrow B^*$ be a function with prefix-closed domain. The *differential of* f is the partial function $D_f\colon A^+ \dashrightarrow B^{(*)}$ defined on $\mathrm{dom}(f)\setminus\{\varepsilon\}$ for all $a \in A, w \in A^*$ by

$$D_f(wa) = \overline{f(w)} \centerdot f(wa). \qquad \triangleleft$$

4.5.6. Lemma. *Let $f\colon A^* \dashrightarrow B^*$ be a function with prefix-closed domain. For all $w = a_1a_2\ldots a_n \in \mathrm{dom}(f)$, $n \geq 1$, we have:*

$$f(w) = f(\varepsilon) \centerdot D_f(a_1) \centerdot D_f(a_1a_2) \centerdot \ldots \centerdot D_f(a_1a_2\ldots a_n), \tag{4.11}$$

$$D_f(w) = D_{f\cdot a_1\ldots a_{n-1}}(a_n), \tag{4.12}$$

$$f(w) = f(\varepsilon) \centerdot D_{f\cdot\varepsilon}(a_1) \centerdot D_{f\cdot a_1}(a_2) \centerdot D_{f\cdot a_1a_2}(a_3) \centerdot \ldots \centerdot D_{f\cdot a_1\ldots a_{n-1}}(a_n). \tag{4.13}$$

Proof. Equation (4.11) holds more or less by definition of D_f:

$$\begin{aligned} f(w) &= f(\varepsilon) \centerdot \overline{f(\varepsilon)} \centerdot f(a_1) \centerdot \overline{f(a_1)} \centerdot f(a_1a_2) \centerdot \ldots \centerdot \overline{f(a_1\ldots a_{n-1})} \centerdot f(a_1\ldots a_n) \\ &= f(\varepsilon) \centerdot D_f(a_1) \centerdot D_f(a_1a_2) \centerdot \ldots \centerdot D_f(a_1a_2\ldots a_n). \end{aligned}$$

To see that equation (4.12) holds, let $v = a_1\ldots a_{n-1}$. We have:

$$\begin{aligned} D_{f\cdot v}(a_n) &= \overline{(f\cdot v)(\varepsilon)} \centerdot (f\cdot v)(a_n) &&= \overline{\overline{f[v]} \centerdot f(v)} \centerdot \overline{f[v]} f(va_n) \\ &= \overline{f(v)} \centerdot f(va_n) &&= D_f(va_n). \end{aligned}$$

Equation (4.13) follows from (4.11) and (4.12). QED

4.5.7. Example. Let f be the behaviour of the step-by-step subsequential transducers from Example 4.5.2. We compute the differential of f. Recall that $f\colon \{a,b\}^* \dashrightarrow \{a,b\}^*$ with $\mathrm{dom}(f) = b^* \cup b^*a$ where $f(b^k) = ba^{k+1}b$ and $f(b^ka) = ba^{k+2}b$ for $k \in \omega$. We have for $k \geq 1$:

$$\begin{aligned} D_f(b^k) &= \overline{f(b^{k-1})} \centerdot f(b^k) &&= \overline{ba^kb} \centerdot ba^{k+1}b &&= \bar{b}ab \quad \text{for } k \geq 1, \\ D_f(b^ka) &= \overline{f(b^k)} \centerdot f(b^ka) &&= \overline{ba^{k+1}b} \centerdot ba^{k+2}b &&= \bar{b}ab \quad \text{for } k \geq 0. \end{aligned} \qquad \triangleleft$$

The analogue of (4.12) for differentials of state behaviour is shown in the following lemma. Note that this lemma is not an immediate consequence of (4.12), since $[\![q]\!]\cdot w \neq [\![d(q)(w)]\!]$ if $d(q)(w)$ is not normalised (cf. Remark 4.4.7), which can be the case in a step-by-step structure.

4.5.8. Lemma. *Let $\mathbb{S} = (Q, o, d, r)$ be a step-by-step subsequential structure, $q \in Q$ and $wa \in \mathrm{dom}([\![q]\!])$ where $w \in A^*$ and $a \in A$, We have: $D_{[\![q]\!]}(wa) = D_{[\![d(q)(w)]\!]}(a)$.*

PROOF. We have:

$$\begin{aligned} D_{[\![q]\!]}(wa) &= \overline{[\![q]\!](w)} \centerdot [\![q]\!](wa) \\ &= \overline{o(q)(w) \centerdot r(d(q)(w))} \centerdot o(q)(w) \centerdot o(d(q)(w))(a) \centerdot r(d(q)(wa)) \\ &= \overline{r(d(q)(w))} \centerdot o(d(q)(w))(a) \centerdot r(d(q)(wa)) \\ &= D_{[\![d(q)(w)]\!]}(a). \end{aligned}$$

QED

Representing behaviour in terms of the differential can be seen as transforming a step-by-step transducer $\mathbb{T}$ into a sequential transducer $\mathbb{T}'$ with prefix which produces output in the free group $B^{(*)}$. A computation in $\mathbb{T}$ corresponds with a computation in $\mathbb{T}'$ as illustrated here:

$$\mathbb{T}\colon \xrightarrow{m} q_0 \overset{r_0}{\Uparrow} \xrightarrow{a_1|u_1} q_1 \overset{r_1}{\Uparrow} \xrightarrow{a_2|u_2} q_2 \quad \cdots \quad q_{n-1} \overset{r_{n-1}}{\Uparrow} \xrightarrow{a_n|u_n} q_n \overset{r_n}{\Uparrow}$$

$$\mathbb{T}'\colon \xrightarrow{m\,r_0} q_0 \xrightarrow{a_1|\overline{r_0}u_1r_1} q_1 \xrightarrow{a_2|\overline{r_1}u_2r_2} q_2 \quad \cdots \quad q_{n-1} \xrightarrow{a_n|\overline{r_{n-1}}u_nr_n} q_n$$

Sequential structures with output in $B^{(*)}$ are not essentially different from sequential structures with output in B^*, and all previously introduced notions for sequential structures apply with identity taken in $B^{(*)}$ where appropriate. This includes extending the transition output function from letters to words, and the definitions of sequential morphisms and behaviour. Differential representations of transducers are almost sequential transducers with output in $B^{(*)}$. The only difference is that differential representations may have a non-trivial initial prefix, whereas in sequential transducers the initial prefix is ε by definition.

4.5.9. DEFINITION. We denote by $\mathsf{Seq}^{(*)}$ the category of sequential structures $\mathbb{S} = (Q, o, d, r)$ in which the transition output function may take values in $B^{(*)}$, i.e., Q, d and r are as in Definition 4.4.11 and $o\colon Q \to (A \dashrightarrow B^{(*)})$. The morphisms of $\mathsf{Seq}^{(*)}$ are the functions which satisfy the conditions in Proposition 4.3.23 by taking equality in $B^{(*)}$ in $(\mathrm{out})_{\mathsf{N}}$.

We denote by $\mathsf{pSeqTra}^{(*)}$ the category which has as its objects subsequential transducers $\mathbb{T} = (\mathbb{S}, i, m)$, where $\mathbb{S}$ is an object in $\mathsf{Seq}^{(*)}$. A morphism in $\mathsf{pSeqTra}^{(*)}$ is a $\mathsf{Seq}^{(*)}$-morphism of the underlying structures which maps initial state to initial state and leaves the initial prefix unchanged, given that the transducers are not empty. ◁

The next definition makes the transformation suggested after Lemma 4.5.8 precise.

4.5.10. Definition (Differential representation). Let $\mathbb{S} = (Q, o, d, r)$ be a step-by-step subsequential structure. The *differential representation of* $\mathbb{S}$ is the object in $\mathsf{Seq}^{(*)}$ denoted by $\mathcal{D}(\mathbb{S}) = (Q, \partial_{\mathbb{S}}, d, \varepsilon)$, where the output function $\partial_{\mathbb{S}} \colon Q \to (\mathbf{1} + B^{(*)})^A$ is defined for $q \in Q$ and $a \in A$ by:

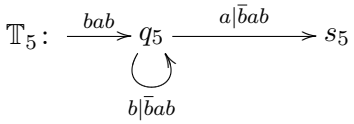
$$\partial_{\mathbb{S}}(q)(a) = \overline{r(q)} \centerdot o(q)(a) \centerdot r(d(q)(a)) \tag{4.14}$$

if $a \in \mathrm{supp}(q)$, and $\star$ otherwise, and $\partial_{\mathbb{S}}(q)(a)$ is reduced in $B^{(*)}$.

For a step-by-step subsequential transducer $\mathbb{T} = (\mathbb{S}, i, m)$ with $\mathbb{S} = (Q, o, d, r)$, we define the *differential representation of* $\mathbb{T}$ as the object in $\mathsf{pSeqTra}^{(*)}$ defined by $\mathcal{D}(\mathbb{T}) = (\mathcal{D}(\mathbb{S}), i, m \centerdot r(i))$. ◁

As with other subscripts, we may leave out $\mathbb{S}$ from $\partial_{\mathbb{S}}$ and simply write ∂ if $\mathbb{S}$ is immaterial, or clear from the context. When we speak of the differential representation of a structure $\mathbb{S}$ or a transducer $\mathbb{T}$, we will always implicitly assume that $\mathbb{S}$ and $\mathbb{T}$ are step-by-step.

4.5.11. Example. It is straightforward to check that the differential representations of the two step-by-step subsequential transducers $\mathbb{T}_3$ and $\mathbb{T}_4$ from Example 4.5.2 are both isomorphic to the object in $\mathsf{pSeqTra}^{(*)}$ depicted below. For example, in $\mathbb{T}_4$, the differential output function at q_4 in b is: $\partial(q_4)(b) = \overline{ab} \centerdot a \centerdot ab = \bar{b}ab$.



◁

Taking differential representations of structures is a map from the objects of Step to the objects of $\mathsf{Seq}^{(*)}$, and as with normalisation, we would like to give a formal argument in the form of a reflectivity result which says the right way of looking at step-by-step structures is in terms of their the differential representation. However, at first glance there is a problem, since $\mathsf{Seq}^{(*)}$ is not a subcategory of Step. The solution to this problem is to also generalise step-by-step structures to produce output in $B^{(*)}$.

4.5.12. Definition. The category $\mathsf{Step}^{(*)}$ has as its objects step-by-step structures $\mathbb{S} = (Q, o, d, r)$ in which the transition output function may take values in $B^{(*)}$, i.e., Q, d and r are as in Definition 4.5.1 and $o \colon Q \to (A \dashrightarrow B^{(*)})$. The morphisms of $\mathsf{Step}^{(*)}$ are the functions which satisfy the characterising conditions for Step-morphism given in Proposition 4.5.3.

The category $\mathsf{StepTra}^{(*)}$ consists of subsequential transducer objects $\mathbb{T} = (\mathbb{S}, i, m)$ where $\mathbb{S}$ is in $\mathsf{Step}^{(*)}$ together with the functions which satisfy the characterising conditions for $\mathsf{StepTra}$-morphisms given in Proposition 4.5.3. ◁

Again, previously defined notions and results, including behaviour and differential representations, all apply unchanged to $\mathsf{Step}^{(*)}$ and $\mathsf{StepTra}^{(*)}$. From the definition of $\mathsf{Seq}^{(*)}$ and $\mathsf{Step}^{(*)}$, and the natural embedding of B^* in $B^{(*)}$, it should be clear that $\mathsf{Seq}^{(*)}$ and Step are full subcategories of $\mathsf{Step}^{(*)}$, and $\mathsf{pSeqTra}^{(*)}$ and $\mathsf{StepTra}$ are full subcategories of $\mathsf{StepTra}^{(*)}$.

We now have a suitable set-up of subcategories, and $\mathcal{D}$ defines an object map from $\mathsf{Step}^{(*)}$ to $\mathsf{Seq}^{(*)}$ and from $\mathsf{StepTra}^{(*)}$ to $\mathsf{pSeqTra}^{(*)}$.

4.5.13. THEOREM. *Let* $\mathbb{S} \in \mathsf{Step}^{(*)}$ *and* $\mathbb{T} \in \mathsf{StepTra}^{(*)}$. *We have:*

1. $id_{\mathbb{S}} \colon \mathbb{S} \to \mathcal{D}(\mathbb{S})$ *is a* $\mathsf{Seq}^{(*)}$*-reflection arrow for* $\mathbb{S}$, *and*
2. $id_{\mathbb{T}} \colon \mathbb{T} \to \mathcal{D}(\mathbb{T})$ *is a* $\mathsf{pSeqTra}^{(*)}$*-reflection arrow for* $\mathbb{T}$.

Hence $\mathsf{Seq}^{(*)}$ *is a reflective subcategory of* $\mathsf{Step}^{(*)}$, *and* $\mathsf{pSeqTra}^{(*)}$ *is a reflective subcategory of* $\mathsf{StepTra}^{(*)}$. *Moreover, by defining* $\mathcal{D}(\alpha) = \alpha$ *for all morphisms* α *in* $\mathsf{Step}^{(*)}$ *and* $\mathsf{StepTra}^{(*)}$, $\mathcal{D}(\_)$ *is a functor* $\mathcal{D} \colon \mathsf{Step}^{(*)} \to \mathsf{Seq}^{(*)}$, *and* $\mathcal{D} \colon \mathsf{StepTra}^{(*)} \to \mathsf{pSeqTra}^{(*)}$.

PROOF. Let $\mathbb{S} = (Q, o, d, r)$ be an object in $\mathsf{Step}^{(*)}$. We first check that $id_{\mathbb{S}}$ is a $\mathsf{Step}^{(*)}$-morphism from $\mathbb{S}$ to $\mathcal{D}(\mathbb{S})$, that is, $id_{\mathbb{S}} = id_Q$ satisfies the conditions given in Proposition 4.5.3 when taking identity in $B^{(*)}$ in $(\mathrm{out})_{\mathsf{S}}$. The conditions (supp) and $(\mathrm{next})_{\mathsf{C}}$ clearly hold for id_Q. In $\mathcal{D}(\mathbb{S})$ the terminal output function is constant equal to ε, hence $(\mathrm{out})_{\mathsf{S}}$ reduces to the requirement that for all $q \in Q$ and $a \in \mathrm{supp}(q)$: $\overline{r(q)} \centerdot o(q)(a) \centerdot r(d(q)(a)) = \partial_{\mathbb{S}}(q)(a)$. This is just the definition of $\partial_{\mathbb{S}}$, hence true. Similarly, condition $(\text{term-out})_{\mathsf{S}}$ reduces to $r(q) \in B^*$ for all $q \in Q$, which also clearly holds.

We must now prove that for any $\mathbb{S}' = (Q', o', d', r') \in \mathsf{Seq}^{(*)}$ and $\alpha \colon \mathbb{S} \to \mathbb{S}'$ in $\mathsf{Step}^{(*)}$, there is a unique $\mathsf{Seq}^{(*)}$-morphism $\alpha' \colon \mathcal{D}(\mathbb{S}) \to \mathbb{S}'$ such that $\alpha = \alpha' \circ id_Q$. As when showing that normalisation is a reflector (Theorem 4.3.25), we will prove that $\alpha' = \alpha$ is the unique choice. To prove that $\alpha \colon \mathcal{D}(\mathbb{S}) \to \mathbb{S}'$ we first note that (supp) and $(\mathrm{next})_{\mathsf{N}}$ are satisfied since the underlying DA's of $\mathcal{D}(\mathbb{S})$ and $\mathbb{S}$ are identical. By the assumption that α is a $\mathsf{Step}^{(*)}$-morphism, α satisfies $(\mathrm{out})_{\mathsf{S}}$, i.e., for all $q \in Q$ and $a \in \mathrm{supp}(q)$ we have: $\partial_{\mathbb{S}}(q)(a) = \overline{r'(q)} \centerdot o'(q)(a) \centerdot r'(d'(q)(a)) = o'(q)(a)$, since $\mathbb{S}' \in \mathsf{Seq}^{(*)}$, hence $(\mathrm{out})_{\mathsf{N}}$ holds and $\alpha \colon \mathcal{D}(\mathbb{S}) \to \mathbb{S}'$ is a $\mathsf{Seq}^{(*)}$-morphism. Uniqueness of α is immediate.

We leave it to the reader to extend the proof for $\mathsf{Step}^{(*)}$ to $\mathsf{pStepTra}^{(*)}$. QED

4.5.14. COROLLARY. *For any* $\mathbb{T}$ *in* $\mathsf{StepTra}$, *we have:* $[\![\mathbb{T}]\!] = [\![\mathcal{D}(\mathbb{T})]\!]$.

PROOF. This is an immediate consequence of Theorem 4.5.13 and the behaviour preservation of subsequential morphisms (which also holds in $\mathsf{StepTra}^{(*)}$). QED

Going to the differential representation only preserves state behaviour modulo an output shift, since differential structures are not normalised.

4.5.15. Lemma. *If $\mathbb{S} = (Q, o, d, r)$ is a step-by-step structure, and $q \in Q$, then: $[\![q]\!]_{\mathbb{S}} = r(q) \centerdot [\![q]\!]_{\mathcal{D}(\mathbb{S})}$.*

Proof. Follows from equation (4.11) and Lemma 4.5.8. Alternatively, one can show that the implicitly defined output shift function for the reflection arrow $id_{\mathbb{S}} \colon \mathbb{S} \to \mathcal{D}(\mathbb{S})$ is $\beta = r$. The result then follows from Proposition 4.3.9. QED

Although state behaviour is not preserved, equivalence in the differential structure captures equivalence of differentials.

4.5.16. Proposition. *If $\mathbb{S} = (Q, o, d, r)$ is a step-by-step structure, and q_1, q_2 are states in Q, then: $D_{[\![q_1]\!]_{\mathbb{S}}} = D_{[\![q_2]\!]_{\mathbb{S}}}$ iff $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$.*

Proof. Since $\mathcal{D}(\mathbb{S})$ is a sequential structure, we always have $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})}(\varepsilon) = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}(\varepsilon) = \varepsilon$. For $w = a_1 \dots a_n \in A^+$, $n \geq 1$, and any $q_0 \in Q$ we have

$$\begin{array}{rcl} [\![q_0]\!]_{\mathcal{D}(\mathbb{S})}(w) & = & \partial_{\mathbb{S}}(q_0)(a_1) \centerdot \partial_{\mathbb{S}}(d(q_0)(a_1))(a_2) \centerdot \dots \centerdot \partial_{\mathbb{S}}(d(q_0)(a_1 \dots a_{n-1})(a_n) \\ & = & D_{[\![q_0]\!]}(a_1) \centerdot D_{[\![d(q_0)(a_1)]\!]}(a_2) \centerdot \dots \centerdot D_{[\![d(q_0)(a_1 \dots a_{n-1})]\!]}(a_n) \\ \text{(Lemma 4.5.8)} & = & D_{[\![q_0]\!]}(a_1) \centerdot D_{[\![q_0]\!]}(a_1 a_2) \centerdot \dots \centerdot D_{[\![q_0]\!]}(a_1 \dots a_n). \end{array}$$

It follows from the above that $D_{[\![q_1]\!]} = D_{[\![q_2]\!]}$ iff $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$. QED

4.5.3 Coalgebras for differentials

In section 4.4, we saw that normalisation yields a coalgebraic representation for subsequential structures. We will now show that differential representations provide an alternative coalgebraic modelling of step-by-step structures which does not go via normalisation.

Objects from $\mathsf{Seq}^{(*)}$ can be modelled as coalgebras in the same way as sequential structures (cf. Proposition 4.4.13) by changing the type functor accordingly. Let the functor $S_0^{(*)} \colon \mathsf{Set} \to \mathsf{Set}$ be defined by:

$$\begin{array}{rcl} S_0^{(*)}(X) & = & (\mathbf{1} + B^{(*)} \times X)^A \times \{\varepsilon\} \\ S_0^{(*)}(f \colon X \to Y) & = & (\mathbf{1} + \mathsf{Id}_{B^{(*)}} \times f)^{\mathsf{Id}_A} \times \{\varepsilon\} \end{array} \tag{4.15}$$

4.5.17. Proposition. *$\mathsf{Seq}^{(*)}$ is isomorphic to $\mathsf{Coalg}({S_0}^{(*)})$.*

Proof. Any structure $\mathbb{S} = (Q, o, d, r)$ in $\mathsf{Seq}^{(*)}$ can be seen as an $S_0^{(*)}$-coalgebra:

$$\langle t, r \rangle \colon Q \to (\mathbf{1} + B^{(*)} \times Q)^A \times \{\varepsilon\}.$$

where the transition structure $t \colon Q \to (\mathbf{1} + B^{(*)} \times Q)^A$ is obtained from $\partial_{\mathbb{S}}$ and d as in (4.6). Checking that the morphisms of $\mathsf{Seq}^{(*)}$ and $\mathsf{Coalg}({S_0}^{(*)})$ coincide is more or less immediate from the definition of $\mathsf{Seq}^{(*)}$-morphisms. QED

The existence of a final object in $\mathsf{Seq}^{(*)}$ does not follow from the existence of a final sequential structure (Theorem 4.4.19), but since ${S_0}^{(*)}$ is a polynomial functor, we know that such a final object exists, and we will see it is straightforward to prove this from first principles. As expected, the final object will have the state behaviours of structures in $\mathsf{Seq}^{(*)}$ as its carrier. Hence we must define output and next-state functions on functions of the type $f\colon A^* \dashrightarrow B^{(*)}$. However, we must do so without the use of the longest common prefix operation, since in $B^{(*)}$ the prefix relation is total. We explain this in detail. A prefix is more generally called a left factor, and for any u, v in $B^{(*)}$, u is a left factor of v if there exists a $w \in B^{(*)}$ such that $v = uw$. But such a w always exists in $B^{(*)}$, just take $w = \overline{u}v$. The totality of the prefix relation also means that all functions of the type $f\colon A^* \dashrightarrow B^{(*)}$, are prefix-preserving. We now adjust the definition of derivative to functions with codomain $B^{(*)}$ and prefix-closed domains (as e.g. differentials). Let $f\colon A^* \dashrightarrow B^{(*)}$ be a function with prefix-closed domain, and let $a \in A$. The derivative of f with respect to a is the partial function $f{\cdot}a\colon A^* \dashrightarrow B^{(*)}$ defined for all $w \in A^*$ by $(f{\cdot}a)(w) = \overline{f(a)} \centerdot f(aw)$ if $aw \in \mathrm{dom}(f)$.

4.5.18. Theorem. *Define*

$$\Psi^{(*)} = \{f\colon A^* \dashrightarrow B^{(*)} \mid \mathrm{dom}(f) \text{ is prefix-closed}, f(\varepsilon) = \varepsilon\}.$$

For $f \in \Psi^{(*)}$ and $a \in A$, define

$$O^{(*)}(f)(a) = f(a), \qquad D^{(*)}(f)(a) = f{\cdot}a, \qquad R^{(*)}(f) = f(\varepsilon).$$

The 4-triple $\mathbf{\Psi}^{(*)} = (\Psi^{(*)}, O^{(*)}, D^{(*)}, R^{(*)})$ is a final object in $\mathsf{Seq}^{(*)}$.

Proof. We first check that $\mathbf{\Psi}^{(*)}$ is well-defined. The output function $O^{(*)}$ takes values in $B^{(*)}$ and the terminal output function $R^{(*)}$ is constant equal to ε on $\Psi^{(*)}$, hence if $\Psi^{(*)}$ is closed under taking derivatives, then we can conclude that $\mathbf{\Psi}^{(*)}$ is a well-defined object in $\mathsf{Seq}^{(*)}$. Let $f \in \Psi^{(*)}$ and $a \in A$. We have $w \in \mathrm{dom}(f{\cdot}a)$ iff $aw \in \mathrm{dom}(f)$, so by the assumption that $\mathrm{dom}(f)$ is prefix-closed, it follows that $\mathrm{dom}(f{\cdot}a)$ is prefix-closed. Moreover, $(f{\cdot}a)(\varepsilon) = \overline{f(a)} \centerdot f(a \centerdot \varepsilon) = \varepsilon$.

We now show that the behaviour map $[\![\_]\!]$ is the final map, i.e., for any $\mathbb{S}$ in $\mathsf{Seq}^{(*)}$, $[\![\_]\!]\colon \mathbb{S} \to \mathbf{\Psi}^{(*)}$ is the unique $\mathsf{Seq}^{(*)}$-morphism. Let $\mathbb{S} = (Q, o, d, r)$ be a sequential structure in $\mathsf{Seq}^{(*)}$. First of all, since all states in $\mathbb{S}$ are final, it is clear that for any $q \in Q$, $\mathrm{dom}([\![q]\!])$ is prefix-closed, hence $[\![q]\!] \in \Psi^{(*)}$. The condition (supp) is easily seen to hold, namely, for $q \in Q$ and $a \in A$ we have: $a \in \mathrm{supp}(q)$ iff $a \in \mathrm{dom}([\![q]\!])$ iff $a \in \mathrm{supp}([\![q]\!])$. Also immediate is the condition $(\mathrm{out})_{\mathsf{N}}$, since for all $q \in Q$ and $a \in \mathrm{supp}(q)$, $[\![q]\!](a) = o(q)(a)$ by definition. Finally, to see that $(\mathrm{next})_{\mathsf{C}}$ holds, we have for all $q \in Q$, $a \in \mathrm{supp}(q)$ and $w \in A^*$:

$$[\![d(q)(a)]\!](w) = \overline{o(q)(a)} \centerdot [\![q]\!](aw) = ([\![q]\!]{\cdot}a)(w).$$

We have thus shown that for any $\mathbb{S}$ in $\mathsf{Seq}^{(*)}$, the map $[\![\_]\!]\colon \mathbb{S} \to \mathbf{\Psi}^{(*)}$ is a $\mathsf{Seq}^{(*)}$-morphism. We leave uniqueness as an exercise to the reader. QED

Since $\mathsf{Seq}^{(*)}$ is reflective in $\mathsf{Step}^{(*)}$ it follows that $\mathbf{\Psi}^{(*)}$ is also a final object in $\mathsf{Step}^{(*)}$. Hence for any step-by-step $\mathbb{S}$, considered as an object in $\mathsf{Step}^{(*)}$, there is a unique $\mathsf{Step}^{(*)}$-morphism from $\mathbb{S}$ to $\mathbf{\Psi}^{(*)}$. By combining the reflectivity of $\mathsf{Seq}^{(*)}$ in $\mathsf{Step}^{(*)}$ with the coalgebraic modelling of $\mathsf{Seq}^{(*)}$ we can now argue that the differential representation is an alternative to normalisation which provides an equally correct way of viewing step-by-step structures as coalgebras.

From Theorem 4.5.18 it also follows that state equivalence is bisimilarity in differential representations.

4.5.19. Corollary. *For any step-by-step structure $\mathbb{S}$, ${S_0}^{(*)}$-bisimilarity is the largest congruence on $\mathcal{D}(\mathbb{S})$ and coincides with state equivalence in $\mathcal{D}(\mathbb{S})$.*

4.5.4 Minimising differential representations

The definition of ${S_0}^{(*)}$-bisimulation amounts to the following. Let $\mathbb{S} = (Q, o, d)$ be an ${S_0}^{(*)}$-coalgebra. A relation $R \subseteq Q \times Q$ is an *${S_0}^{(*)}$-bisimulation* on $\mathbb{S}$, if for all $\langle q, s\rangle \in R$:

(d0) $\mathrm{supp}(q) = \mathrm{supp}(s)$;

(d1) for all $a \in \mathrm{supp}(q)$: $o(q)(a) = o(s)(a)$ (in the free group $B^{(*)}$); and

(d2) for all $a \in \mathrm{supp}(q)$: $\langle d(q)(a), d(s)(a)\rangle \in R$.

Given a finite ${S_0}^{(*)}$-coalgebra $\mathbb{S} = (Q, o, d)$, we can determine state equivalence on $\mathbb{S}$ by using a small variation on the algorithm from subsection 4.4.3 for computing state equivalence in normalised structures. The idea is again to perform the refinement algorithm for DFA's, but this time we take as the initial partition, the largest equivalence relation P_0^d on Q which satisfies (d0) and (d1).

4.5.20. Lemma. *Let $\mathbb{S} = (Q, o, d)$ be a finite ${S_0}^{(*)}$-coalgebra. We can compute the largest equivalence relation P_0^d on Q which satisfies (d0) and (d1) in time $O(\|o\|\,|A||Q|\log(|Q|))$ where $\|o\| := \max\{|o(q)(a)| \mid q \in Q, a \in A\}$.*

Proof. P_0^d can be computed in essentially the same way as P_0^s, so we only provide a sketch and refer to Lemma 4.4.9 for details. We again use a binary seach tree, but now a state q is inserted with key value $c(q) := \langle o(q)(a_1), \ldots, o(q)(a_k)\rangle$ where $|A| = k$. In order to define a linear ordering on key values, it suffices to define a linear ordering on $B \cup \{\overline{b} \mid b \in B\}$, since we can then extend this ordering lexicographically to reduced elements of $B^{(*)}$ and key values as in Lemma 4.4.9. As before, we obtain a linear ordering $b_1 \prec b_2 \prec \ldots \prec b_n$ on $B = \{b_1, b_2, \ldots, b_n\}$

by enumeration, and we extend $\prec$ to $B \cup \{\overline{b} \mid b \in B\}$ by defining $\overline{b} \prec \overline{b'}$ iff $b \prec b'$ and $\overline{b} \prec b'$ for all $b, b' \in B$, i.e., $\overline{b_1} \prec \overline{b_2} \prec \ldots \prec \overline{b_n} \prec b_1 \prec b_2 \prec \ldots \prec b_n$. The size of $c(q)$-values is now $O(|A|\|o\|)$ which yields a time complexity of $O(\|o\|\,|A|\,|Q|\log(|Q|))$ for inserting all $(c(q), q)$ pairs into the tree. QED

Lemma 4.5.20 can be used to give a bound on the time complexity of computing state equivalence on $\mathcal{D}(\mathbb{S})$ starting with a finite step-by-step structure $\mathbb{S}$.

4.5.21. PROPOSITION. *Let $\mathbb{S} = (Q, o, d, r)$ be a finite step-by-step structure. Computing $\mathcal{D}(\mathbb{S})$ and state equivalence on $\mathcal{D}(\mathbb{S})$ can be done in time:*

Compute $\partial_{\mathbb{S}}$:	$O(M\|r\|)$,
Compute state equivalence on $\mathcal{D}(\mathbb{S})$:	$O((2\|r\| + \|o\|)\,\|A\|\,\|Q\|\log(\|Q\|))$

where M is the number of transitions in $\mathbb{S}$, $\|r\| := \max\{|r(q)| \mid q \in Q\}$ and $\|o\| := \max\{|o(q)(a)| \mid q \in Q, a \in A\}$.

PROOF. The time to compute a single value $\partial_{\mathbb{S}}(q)(a)$ for $q \in Q$ and $a \in \mathrm{supp}(q)$ is proportional to the time it takes to reduce $\overline{r(q)} \cdot o(q)(a) \cdot r(d(q)(a))$ using applications of the $B^{(*)}$-identity $\overline{x}x = \varepsilon$. The maximal number of these reduction steps is bounded by $\|r\|$. Hence computing all $\partial_{\mathbb{S}}$-values can be done in time $O(M\|r\|)$. From Lemma 4.5.20 we know that we can compute the initial partition P_0^d on $\mathcal{D}(\mathbb{S})$ in time $O(\|\partial_{\mathbb{S}}\|\,|A|\,|Q|\log(|Q|))$. From the definition of $\partial_{\mathbb{S}}$, we have that $\|\partial_{\mathbb{S}}\| \leq 2\|r\| + \|o\|$. Hence P_0^d can be computed in time $O((2\|r\| + \|o\|)\,|A|\,|Q|\log(|Q|))$. The refinement part of the algorithm can be done in time $O(|A||Q|\log(|Q|))$ (cf. [74]). Adding up the time needed for computing P_0^d and the time needed for refinement (under the big-O), we find that state equivalence can be computed in time $O((2\|r\| + \|o\|)\,|A|\,|Q|\log(|Q|))$. QED

Proposition 4.5.21 gives us a method to decide equivalence of step-by-step transducers via differentials without normalisation. We will use the fact that the disjoint union of two step-by-step structures is again a step-by-step structure. The *disjoint union* of two step-by-step structures $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1))$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$ is the step-by-step structure $\mathbb{S}_1 + \mathbb{S}_2 = (Q, o, d, r)$ where $Q = Q_1 + Q_2$ (i.e., the disjoint union of the sets Q_1 and Q_2), and for $q \in Q_j$, $j \in \{1, 2\}$, and $a \in A$, $o(q)(a) = o_j(q)(a)$, $d(q)(a) = d_j(q)(a)$ and $r(q) = r_j(q)$. Formally, $\mathbb{S}_1 + \mathbb{S}_2$ is the coproduct of $\mathbb{S}_1$ and $\mathbb{S}_2$ in $\mathsf{Coalg}(S)$, which implies that the inclusion maps $\iota_j \colon Q_j \to Q$ are S-coalgebra morphisms, and hence $[\![q]\!]_{\mathbb{S}_1+\mathbb{S}_2} = [\![q]\!]_{\mathbb{S}_j}$, if $q \in Q_j$ for $j \in \{1, 2\}$. This can also be proved directly by using the definition of behaviour. Furthermore, since S-coalgebra morphisms are in particular subsequential morphisms, the inclusion maps are also morphisms in Step, and for any $j \in \{1, 2\}$: $\iota_j \colon \mathcal{D}(\mathbb{S}_1) \to \mathcal{D}(\mathbb{S}_1 + \mathbb{S}_2)$ is a morphism in $\mathsf{Seq}^{(*)}$.

Since $\mathsf{Seq}^{(*)}$-morphisms preserve behaviour, we find that for any $j \in \{1,2\}$, if $q \in Q_j$:

$$[\![q]\!]_{\mathcal{D}(\mathbb{S}_j)} = [\![q]\!]_{\mathcal{D}(\mathbb{S}_1+\mathbb{S}_2)}. \tag{4.16}$$

4.5.22. Theorem. *Let $\mathbb{T}_1 = (\mathbb{S}_1, i_1, m_1)$ and $\mathbb{T}_2 = (\mathbb{S}_2, i_2, m_2)$ be two step-by-step transducers, where $\mathbb{S}_1 = (Q_1, o_1, d_1, r_1))$ and $\mathbb{S}_2 = (Q_2, o_2, d_2, r_2)$. We have: $\mathbb{T}_1$ and $\mathbb{T}_2$ are equivalent if and only if*

$$m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2) \quad \text{and} \quad [\![i_1]\!]_{\mathcal{D}(\mathbb{S}_1+\mathbb{S}_2)} = [\![i_2]\!]_{\mathcal{D}(\mathbb{S}_1+\mathbb{S}_2)}.$$

We can decide whether $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$ in time: $O((2\|r\| + \|o\|)|A|N\log(N))$, where $N = |Q_1| + |Q_2|$, $\|r\| = \max\{|r_j(q_j)| \mid q_j \in Q_j, j \in \{1,2\}\}$, $\|o\| = \max\{|o_j(q_j)(a)| \mid q_j \in Q_j, j \in \{1,2\}, a \in A\}$.

Proof. Let $j \in \{1,2\}$. From the definition of $\mathcal{D}$, the initial prefix of $\mathcal{D}(\mathbb{T}_j)$ is $m_j \centerdot r_j(i_j)$, and since $\mathcal{D}$ preserves transducer behaviour (Corollary 4.5.14), we find that: $[\![\mathbb{T}_j]\!] = [\![\mathcal{D}(\mathbb{T}_j)]\!] = m_j \centerdot r_j(i_j) \centerdot [\![i_j]\!]_{\mathcal{D}(\mathbb{S}_j)}$. By equation (4.16), we have for any $q \in Q_j$ that $[\![q]\!]_{\mathcal{D}(\mathbb{S}_j)} = [\![q]\!]_{\mathcal{D}(\mathbb{S}_1+\mathbb{S}_2)}$. Hence it follows that $[\![\mathbb{T}_1]\!] = [\![\mathbb{T}_2]\!]$ if and only if $m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2)$ and $[\![i_1]\!]_{\mathcal{D}(\mathbb{S}_1+\mathbb{S}_2)} = [\![i_2]\!]_{\mathcal{D}(\mathbb{S}_1+\mathbb{S}_2)}$. The desired decision method is obtained by testing $m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2)$, computing $\partial_{\mathbb{S}_1+\mathbb{S}_2}$, and computing state equivalence on $\mathcal{D}(\mathbb{S}_1 + \mathbb{S}_2)$. We will ignore the time needed to check $m_1 \centerdot r_1(i_1) = m_2 \centerdot r_2(i_2)$, since we assume the other two computations will dominate the time cost. The time complexity bound now follows from Proposition 4.5.21. The costs of computing $\partial_{\mathbb{S}_1+\mathbb{S}_2}$ is $O(M\|r\|)$, where M is the number of transitions in $\mathbb{S}_1 + \mathbb{S}_2$, but since $M \leq |A|N$, we find that the state equivalence computation dominates the time complexity of the entire decision method. QED

We now have two ways of constructing a minimal representation of a step-by-step transducer $\mathbb{T}$. One is quotienting $\mathcal{N}(\mathbb{T})$ with S-bisimilarity, the other is quotienting $\mathcal{D}(\mathbb{T})$ with ${S_0}^{(*)}$-bisimilarity. In Theorem 4.5.26 below we will show that minimising the differential representation $\mathcal{D}(\mathbb{T})$ yields the differential representation of the minimisation of $\mathbb{T}$. In order to prove this, we first make an easy, but useful observation, that links differentials and normalisation.

4.5.23. Lemma. *If $\mathbb{S}_1, \mathbb{S}_2$ are in Step, and $\alpha\colon \mathbb{S}_1 \to \mathbb{S}_2$ is a subsequential morphism, then for all states q in $\mathbb{S}_1$ and all $a \in A$: $\partial_{\mathbb{S}_1}(q)(a) = \partial_{\mathbb{S}_2}(\alpha(q))(a)$. In particular, since $id_{\mathbb{S}}\colon \mathbb{S} \to \mathcal{N}(\mathbb{S})$ is a subsequential morphism, $\partial_{\mathbb{S}}(q)(a) = \partial_{\mathcal{N}(\mathbb{S})}(q)(a)$ for all $q \in Q$ and $a \in A$, which implies that $\mathcal{D}(\mathbb{S}) = \mathcal{D}(\mathcal{N}(\mathbb{S}))$.*

Proof. Follows from the fact that $\alpha\colon \mathbb{S}_1 \to \mathbb{S}_2$ in Step implies that $\alpha\colon \mathcal{D}(\mathbb{S}_1) \to \mathcal{D}(\mathbb{S}_2)$ in $\mathsf{Seq}^{(*)}$ (Theorem 4.5.13), and that $\mathsf{Seq}^{(*)}$-morphisms satisfy the condition $(\mathrm{out})_{\mathsf{N}}$ (cf. page 85). QED

We can now show that for a step-by-step transducer $\mathbb{T}$, the state equivalence relations on $\mathcal{D}(\mathbb{T})$ and $\mathcal{N}(\mathbb{T})$ are identical (as relations on the state set).

4.5.24. Proposition. *Let $\mathbb{S}$ be a step-by-step subsequential structure. For all states q_1 and q_2 in $\mathbb{S}$: $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$ iff $[\![q_1]\!]_{\mathcal{N}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{N}(\mathbb{S})}$.*

Proof. Let $\mathbb{S} = (Q, o, d, r)$, $\mathcal{D}(\mathbb{S}) = (Q, \partial_{\mathbb{S}}, d)$ and $\mathcal{N}(\mathbb{S}) = (Q, o', d, r')$. Note that since all three structures have the same underlying DA, we have for all $q \in Q$, $\mathrm{dom}([\![q]\!]_{\mathbb{S}}) = \mathrm{dom}([\![q]\!]_{\mathcal{D}(\mathbb{S})}) = \mathrm{dom}([\![q]\!]_{\mathcal{N}(\mathbb{S})})$. Using the fact that $(id_Q, \hat{\beta}) \colon \mathbb{S} \to \mathcal{N}(\mathbb{S})$ is a subsequential morphism (Theorem 4.3.25), we get from Proposition 4.3.9 and Lemma 4.5.15 for all $q \in Q$:

$$[\![q]\!]_{\mathcal{N}(\mathbb{S})} = \overline{\hat{\beta}(q)} \centerdot [\![q]\!]_{\mathbb{S}} = \overline{\hat{\beta}(q)} \centerdot r(q) \centerdot [\![q]\!]_{\mathcal{D}(\mathbb{S})} = r'(q) \centerdot [\![q]\!]_{\mathcal{D}(\mathbb{S})}. \tag{4.17}$$

First, if $[\![q_1]\!]_{\mathcal{N}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{N}(\mathbb{S})}$, then in particular, $[\![q_1]\!]_{\mathcal{N}(\mathbb{S})}(\varepsilon) = [\![q_2]\!]_{\mathcal{N}(\mathbb{S})}(\varepsilon)$, i.e., $r'(q_1) = r'(q_2)$, and hence from (4.17) we get that $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$.

Now assume that $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$. From (4.17) it follows that in order to show that $[\![q_1]\!]_{\mathcal{N}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{N}(\mathbb{S})}$, it suffices to prove that $r'(q_1) = r'(q_2)$. Suppose first that $\mathrm{supp}(q_1) = \mathrm{supp}(q_2) = \emptyset$. In this case, $\hat{\beta}(q_1) = r(q_1)$ and hence $r'(q_1) = \overline{\hat{\beta} q_1} \centerdot r(q_1) = \varepsilon$. Similarly, we get $r'(q_2) = \varepsilon$, and so $r'(q_1) = r'(q_2)$. Now suppose $\mathrm{supp}(q_1) = \mathrm{supp}(q_2) \neq \emptyset$. Since $\mathcal{N}(\mathbb{S})$ is normalised, there must be $a_1, a_2 \in A$ such that $\mathrm{lcp}(\{r'(q_1), o'(q_1)(a_1)\}) = \mathrm{lcp}(\{r'(q_2), o'(q_2)(a_2)\}) = \varepsilon$. By assumption, $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})} = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}$, hence in particular: $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})}(a_1) = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}(a_1)$, that is, $\partial_{\mathbb{S}}(q_1)(a_1) = \partial_{\mathbb{S}}(q_2)(a_1)$. From Lemma 4.5.23 we know that $\partial_{\mathbb{S}}(q_j)(a_1) = \partial_{\mathcal{N}(\mathbb{S})}(q_j)(a_1)$, for $j \in \{1, 2\}$, and hence $\partial_{\mathcal{N}(\mathbb{S})}(q_1)(a_1) = \partial_{\mathcal{N}(\mathbb{S})}(q_2)(a_1)$, that is:

$$\overline{r'(q_1)} \centerdot o'(q_1)(a_1) \centerdot r'(d(q_1)(a_1)) = \overline{r'(q_2)} \centerdot o'(q_2)(a_1) \centerdot r'(d(q_2)(a_1)). \tag{4.18}$$

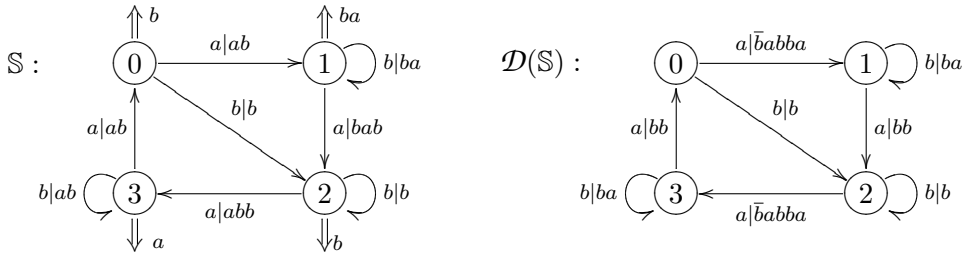
Letting $v = \mathrm{lcp}(\{r'(q_2), o'(q_2)(a_1) \centerdot r'(d(q_2)(a_1))\})$ it follows from the assumption on a_1 and (4.18) that $r'(q_2) = v \centerdot r'(q_1)$. Now we use that $[\![q_1]\!]_{\mathcal{D}(\mathbb{S})}(a_2) = [\![q_2]\!]_{\mathcal{D}(\mathbb{S})}(a_2)$ and with the same arguments we used to reach (4.18) we get:

$$\begin{aligned} \overline{r'(q_1)} \centerdot o'(q_1)(a_2) \centerdot r'(d(q_1)(a_2)) &= \overline{r'(q_2)} \centerdot o'(q_2)(a_2) \centerdot r'(d(q_2)(a_2)) \\ &= \overline{r'(q_1)} \centerdot \overline{v} \centerdot o'(q_2)(a_2) \centerdot r'(d(q_2)(a_2)). \end{aligned} \tag{4.19}$$

Since $v \preceq r'(q_2)$ we have by our choice of a_2 that $\mathrm{lcp}(\{v, o'(q_2)(a_2)\}) = \varepsilon$. Hence from (4.19) we can now conclude that $v = \varepsilon$ and hence $r'(q_1) = r'(q_2)$. QED

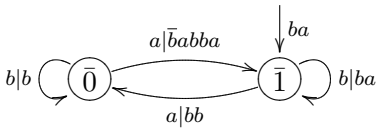
The next example illustrates the result of Proposition 4.5.24 and the difference between the two types of minimal structures.

4.5.25. Example. Consider the following transition diagram of a step-by-step subsequential structure $\mathbb{S} = (Q, o, d, r)$, and its differential representation $\mathcal{D}(\mathbb{S})$:

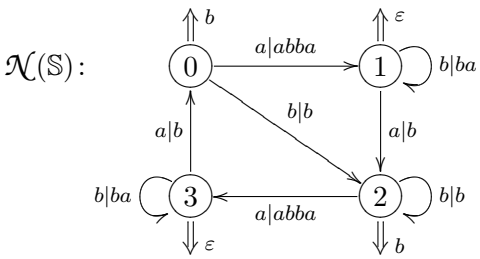


For $i \in \{0,1,2,3\}$, let $f_i = [\![i]\!]_{\mathbb{S}}$. We find that: $f_0(\varepsilon) = r(0) = b$, $f_1(\varepsilon) = r(1) = ba$, $f_2(\varepsilon) = r(2) = b$, and $f_3(\varepsilon) = r(3) = a$. It can easily be checked that the relation $R = \{\langle 0,2\rangle; \langle 1,3\rangle; \langle 0,0\rangle; \langle 1,1\rangle; \langle 2,2\rangle; \langle 3,3\rangle\}$ is the maximal ${S_0}^{(*)}$-bisimulation on $\mathcal{D}(\mathbb{S})$. Proposition 4.5.16 then tells us that $D_{f_0} = D_{f_2}$ and $D_{f_1} = D_{f_3}$. Furthermore, since $f_0(\varepsilon) = f_2(\varepsilon)$, we can conclude (from equation (4.11) in Lemma 4.5.6) that $f_0 = f_2$. We can obtain a minimal sequential transducer with output in $B^{(*)}$ which computes f_1 by quotienting $\mathcal{D}(\mathbb{S})$ with R and initialising this structure with the R-class containing 1, and adding initial prefix $f_1(\varepsilon) = ba$. (Similarly for the functions f_0, f_2 and f_3): Let $\bar{0} = \{0,2\}$ and $\bar{1} = \{1,3\}$.

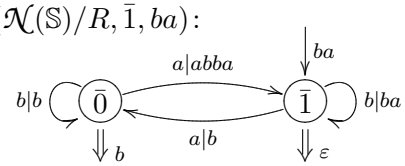
$(\mathcal{D}(\mathbb{S})/R, \bar{1}, ba)$:



Alternatively, we could compute and minimise $\mathcal{N}(\mathbb{S})$. It can easily be verified that: $\hat{\beta}(0) = \varepsilon$, $\hat{\beta}(1) = ba$, $\hat{\beta}(2) = \varepsilon$, $\hat{\beta}(3) = a$. $\mathcal{N}(\mathbb{S})$ is illustrated below on the left. Again, it is easy to confirm that the state equivalence relation on $\mathcal{N}(\mathbb{S})$ is equal to R, and we now obtain a minimal normalised subsequential transducer with behaviour f_1 by quotienting $\mathcal{N}(\mathbb{S})$ with R, initialising with the R-class $\bar{1}$ and adding the initial prefix $\hat{\beta}(1) = ba$.



$(\mathcal{N}(\mathbb{S})/R, \bar{1}, ba)$:



$\triangleleft$

From Proposition 4.5.24 we now know that given a step-by-step structure $\mathbb{S} = (Q, o, d, r)$, ${S_0}^{(*)}$-bisimilarity on $\mathcal{D}(\mathbb{S})$ and S-bisimilarity on $\mathcal{N}(\mathbb{S})$ coincide as relations on Q. We will denote this relation by $\equiv$, and we let $(\_)_\equiv \colon Q \to Q/\!\equiv$ be the quotient map which sends a state $q \in Q$ to its $\equiv$-class $q_\equiv \in Q/\!\equiv$. Since $\equiv$ is a congruence on $\mathcal{D}(\mathbb{S})$ and $\mathcal{N}(\mathbb{S})$ the quotient map is a $\mathsf{Seq}^{(*)}$-morphism $(\_)_\equiv \colon \mathcal{D}(\mathbb{S}) \to \mathcal{D}(\mathbb{S})/\!\equiv$, and a subsequential morphism $(\_)_\equiv \colon \mathcal{N}(\mathbb{S}) \to \mathcal{N}(\mathbb{S})/\!\equiv$ in Step. We can now state and prove the exact relationship between the two minimal representations.

4.5.26. Theorem. *Let $\mathbb{S}$ be a step-by-step subsequential structure, and $\mathbb{T}$ a step-by-step subsequential transducer. We have:*

$$\mathcal{D}(\mathcal{N}(\mathbb{S})/\!\equiv) \;=\; \mathcal{D}(\mathbb{S})/\!\equiv \qquad \textit{and} \qquad \mathcal{D}(\mathcal{N}(\mathbb{T})/\!\equiv) \;=\; \mathcal{D}(\mathbb{T})/\!\equiv .$$

Proof. We first show the result for structures. Let $\mathbb{S}$ be a step-by-step structure, and assume $\mathcal{D}(\mathbb{S})/\!\equiv$ is equal to $(Q/\!\equiv, \partial_\equiv, d_\equiv)$. Since $\mathcal{D}$ and $\mathcal{N}$ do not change the underlying DA, the result follows once we show that $\partial_\equiv$ equals $\partial_{\mathcal{N}(\mathbb{S})/\equiv}$. But this is almost immediate: The quotient map is a $\mathsf{Seq}^{(*)}$-morphism $(\_)_\equiv \colon \mathcal{D}(\mathbb{S}) \to \mathcal{D}(\mathbb{S})/\!\equiv$ hence for all $q \in Q$ and $a \in \mathrm{supp}(q)$: $\partial_\equiv(q_\equiv)(a) = \partial_{\mathbb{S}}(q)(a)$, and from Lemma 4.5.23, we get that for all $q \in Q$ and $q \in \mathrm{supp}(q)$: $\partial_{\mathbb{S}}(q)(a) = \partial_{\mathcal{N}(\mathbb{S})}(q)(a)$. Finally, since the quotient map is a subsequential morphism $(\_)_\equiv \colon \mathcal{N}(\mathbb{S}) \to \mathcal{N}(\mathbb{S})/\!\equiv$ in Step, we have again from Lemma 4.5.23 that $\partial_{\mathcal{N}(\mathbb{S})}(q)(a) = \partial_{\mathcal{N}(\mathbb{S})/\equiv}(q_\equiv)(a)$, for all $q \in Q$ and $a \in \mathrm{supp}(q)$. The proof for the transducer case follows from result for structures as soon as we can show that $\mathcal{D}(\mathbb{T})$ and $\mathcal{D}(\mathcal{N}(\mathbb{T}))$ have the same initial prefix. Let $\mathbb{T} = (Q, o, d, r, i, m) \in \mathsf{StepTra}$. The initial prefix of $\mathcal{D}(\mathbb{T})$ is $m \centerdot r(i)$ (Definition 4.5.10). Letting m' and r' denote the initial prefix and the terminal output function in $\mathcal{N}(\mathbb{T})$, the initial prefix in $\mathcal{D}(\mathcal{N}(\mathbb{T}))$ is $m' \centerdot r'(i) = m \centerdot \hat{\beta}_{\mathbb{T}}(i) \centerdot r'(i) = m \centerdot r(i)$ (cf. Definition 4.3.21). QED

We end this section with a discussion on the advantages and disadvantages of using the differential representation for deciding equivalence of step-by-step transducers. Let us refer to the method described in Theorem 4.5.22 and Proposition 4.5.21 for deciding equivalence of step-by-step transducers as *equivalence-via-differentials*. The classic method which determines equivalence by first normalising and then computing state equivalence on the normalisations, we will refer to as *equivalence-via-normalisation*. In the rest of this section, assume $\mathbb{S}$ is a finite step-by-step structure.

Comparing the time complexities for the state equivalence computations in both methods, $O((|A|\,\|o\| + \|r\|)|Q|\log(|Q|))$ (over $\mathcal{N}(\mathbb{S})$, cf. Proposition 4.4.10) and $O((|A|\,(\|o\| + 2\|r\|))|Q|\log(|Q|))$ (over $\mathcal{D}(\mathbb{S})$, cf. Proposition 4.5.21), we see that there is some reason to expect that deciding state equivalence on $\mathcal{N}(\mathbb{S})$ is more efficient than deciding state equivalence on $\mathcal{D}(\mathbb{S})$. In particular, if terminal

output labels are much longer than the output labels generated on transitions, then the size of the data records which must be compared when computing the initial partition, will be much larger for $\mathcal{D}(\mathbb{S})$ than for $\mathcal{N}(\mathbb{S})$. However, for structures in which the length of output labels is small with respect to the number of states, the two state equivalence computations could display similar running times.

More interestingly, from our point of view, is the comparison between the computations of $\mathcal{D}(\mathbb{S})$ and $\mathcal{N}(\mathbb{S})$. Using Béal & Carton's normalisation algorithm from [17], $\mathbb{S}$ can be normalised in time $O((\|\hat{\beta}\| + 1)M)$, where $\|\hat{\beta}\|$ is the length of the longest $\hat{\beta}(q)$-value, and M is the number of transitions in $\mathbb{S}$. In Proposition 4.5.21, we saw that $\partial_{\mathbb{S}}$ can be computed in time $O(M\|r\|)$. In terms of big-O complexity, there seems to be no advantage in using equivalence-via-differentials. However, Béal & Carton's normalisation algorithm is a global computation in which the entire structure is inspected and updated through a depth-first search in each iteration. Moreover, the algorithm requires a preprocessing step where the strongly connected components of $\mathbb{S}$ are determined, and in order to achieve the reported complexity, a number of data structures are needed which, in our opinion, makes it more difficult to understand the correctness of the algorithm. In contrast, the computation of $\partial_{\mathbb{S}}$-values is *local*, in the sense that for each state q and $a \in \mathrm{supp}(q)$, we only need to know the values of $r(q), o(q)(a)$ and $r(d(q)(a))$ in order to determine $\partial_{\mathbb{S}}(q)(a)$, and this computation can be done using simple, intuitive list operations and still achieves a similar time complexity.

We also observe that it is not necessary to store an explicit representation of $\mathcal{D}(\mathbb{S})$ in order to start computing state equivalence on $\mathcal{D}(\mathbb{S})$. From a representation of $\mathbb{S}$, it is possible to compute the records $c(q) = \langle \partial_{\mathbb{S}}(q)(a_1), \ldots, \partial_{\mathbb{S}}(q)(a_k) \rangle$ one by one just before inserting them into the binary search tree which is used in the computation of the initial partition P_0^d (cf. proof of Lemma 4.5.20). If we want to construct the actual minimal differential representation, then we can retrieve the $\partial_{\mathbb{S}}$-values from the tree. In case many states turn out to be equivalent with respect to P_0^d, i.e., the number of P_0^d-classes is much smaller than the number of states $|Q|$, some space efficiency should be gained by only storing $\partial_{\mathbb{S}}$-values for each P_0^d-class, rather than for each state. For very large state spaces, this local nature of the differential could be interesting, since the computation of $\partial_{\mathbb{S}}$ can be divided into smaller batches which can be processed in sequence or parallel.

The obvious shortcoming of the equivalence-via-differentials method is that it only works for step-by-step transducers, and not for subsequential transducers in general. Moreover, an actual implementation of a step-by-step behaviour $f\colon A^* \dashrightarrow B^*$ should be based on the minimal, normalised representation $\mathbb{T}_f$, since the output generated by the (minimal) differential representation of f would have to be reduced in the free group in order to reconstruct the f-value.

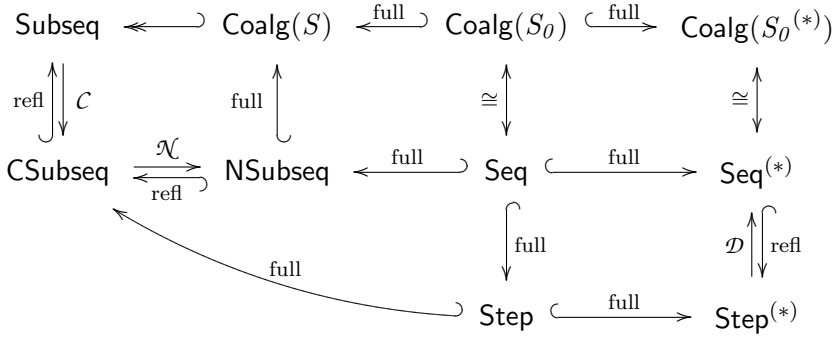
But given the fact that equivalence-via-differentials can be implemented in a straightforward manner which for large state spaces should display a performance comparable with equivalence-via-normalisation, we believe that for step-by-step transducers equivalence-via-differentials could be an interesting alternative.

4.6 Conclusion

Although subsequential structures as objects have the type of coalgebras for a functor $S\colon \mathsf{Set} \to \mathsf{Set}$ they can, in general, not be seen as S-coalgebras, since their word function semantics requires a notion of morphism which is more general than the notion of S-coalgebra morphism. However, the results of this chapter show that normalisation and taking differentials are a form of coalgebraisation. This is made precise by showing that normalised subsequential structures form a full subcategory of the category $\mathsf{Coalg}(S)$ of all S-coalgebras (Proposition 4.4.2), and taking differentials can be seen as a functor $\mathcal{D}$ from the category of step-by-step subsequential structures to the category $\mathsf{Coalg}({S_0}^{(*)})$ for a functor ${S_0}^{(*)}\colon \mathsf{Set} \to \mathsf{Set}$ (Definition 4.5.10 and Proposition 4.5.17). The coalgebraic nature of normalised structures explains why state equivalence in normalised structures can be computed much in the same way as it is done for deterministic finite automata. State equivalence in differential representations of step-by-step structures can also be determined in a similar way, and the corresponding quotient construction gives rise to an alternative form of minimal representation. We have provided a detailed description of how one can adapt the known method for DFA-minimisation to normalised structures (Proposition 4.4.10) and differential representations (Proposition 4.5.21). For the purpose of deciding equivalence of step-by-step transducers, we believe that the decision method obtained by computing state equivalence on the differential structure is an interesting alternative to computing state equivalence on the normalisation. This claim is based on the straightforward, local manner in which the differential can be computed, as opposed to the more complicated and global nature of known normalisation algorithms (cf. [37, 17]).

We also showed that normalisation and taking differentials are functorial, in fact, these operations are reflectors $\mathcal{N}$ and $\mathcal{D}$, respectively (Theorems 4.3.25 and 4.5.13). These results provide an argument for saying that the right way of thinking about subsequential structures is in their coalgebraic, normalised form. For step-by-step structures, the differential representation yields an alternative, but equally correct, coalgebraic description. In the diagram below we provide an overview of the relationships between the various classes of subsequential structures and coalgebras that have been studied in this chapter. The inclusion arrows indicate embeddings of categories; a double-headed arrow indicates that

the embedding is surjective on objects; and the labels 'full', 'refl' and $\cong$ indicate whether the embedding is full, reflective or an isomorphism, respectively.



As directions for future research, we mention that in automatic speech recognition, subsequential transducers with weighted transitions play an important role (cf. [99]). Since weighted or probabilistic systems can be modelled coalgebraically [16], one could try to extend the current coalgebraic modelling to weighted subsequential transducers. On the side of formal languages and transductions, the natural next step would be to try to give a coalgebraic modelling of sequential bimachines and rational functions [41]. We expect this to be a non-trivial exercise, if at all possible, since a unique canonical minimal bimachine seems not to exist [123].

The problems we encountered in the coalgebraic modelling of arbitrary subsequential structures arise from the presence of internal states and the fact that the word function semantics equates output in the free monoid. These issues could perhaps be dealt with by extending the coalgebraic setting from being purely set based, to one in which the monoid identities are formally included. Another idea would be to look for alternative equivalence notions along the lines of weak bisimilarity. However, weak bisimilarity in coalgebras is not very well understood, but some results may be found in [125, 127, 142]. In this context, we also mention that although subsequential structures seem to allow a kind of internal steps, these steps are not entirely unobservable since an input letter is always consumed. Moreover, if we can find the right setting for a coalgebraic modelling of subsequential transducers, we may also gain insights into the possibility of capturing the semantics of Büchi automata coalgebraically. To be a little more explicit, we can think of a deterministic Büchi automaton $\mathbb{B}$ as a machine which outputs a 1 on transitions leaving a final state, and the empty word on all other transitions. The language accepted by $\mathbb{B}$ then consists of the infinite input sequences which are mapped to the infinite sequence of 1's. An adequate formalisation should take into account that two deterministic Büchi automata can be equivalent without generating the same input on all finite prefixes of the input sequence.

Another direction for future research would be to find out whether existing coalgebraic specification languages (cf. [28, 86, 111]) are useful for expressing properties in the application domains of subsequential transducers. For example, the regular expressions for polynomial coalgebras given in [28] provide an expressive formal language for specifying normalised and step-by-step subsequential transducers. The question is whether this language is suitable for expressing properties of natural language processing (cf. [99, 98]). We would also like to know if our results on step-by-step transducers and differential representations are practically useful in this or other application domains.

Finally, we mention that step-by-step transducers are implicitly used in [29] to give a proof of Choffrut's characterisation theorem for subsequential functions. It would be interesting to see if Choffrut's result can be reinterpreted in the coalgebraic setting presented here.

Chapter 5

Bisimilarity in neighbourhood structures

5.1 Introduction

Neighbourhood semantics [35] forms a generalisation of Kripke semantics, and it has become the standard tool for reasoning about non-normal modal logics in which (Kripke valid) principles such as $\Box p \wedge \Box q \rightarrow \Box(p \wedge q)$ and $\Box p \rightarrow \Box(p \vee q)$ are considered not to hold. In a neighbourhood model, with each state one associates a collection of subsets of the universe (called its neighbourhoods), and a modal formula $\Box\varphi$ is true at a state s if the truth set of φ is a neighbourhood of s. The modal logic of all neighbourhood models is called classical modal logic.

Neighbourhood semantics was invented in 1970 by Scott and Montague (independently in [140] and [100]); and Segerberg [141] presents some basic results about neighbourhood models and the classical modal logics that correspond to them. These and other salient results were incorporated by Chellas in his textbook [35]. During the past 15-20 years, non-normal modal logics have emerged in the areas of computer science and social choice theory, where system (or agent) properties are formalised in terms of various notions of ability in strategic games (e.g. [6, 115]). These logics have in common that they are monotonic, meaning they contain the above-mentioned formula $\Box p \rightarrow \Box(p \vee q)$. The corresponding property of neighbourhood models is that neighbourhood collections are closed under supersets. Non-monotonic modal logics occur in deontic logic (see e.g. [46]) where monotonicity can lead to paradoxical obligations, and in the modelling of knowledge and related epistemic notions (cf. [148, 108]). Furthermore, the topological semantics of modal logic can be seen as neighbourhood semantics (see [32] and references).

Neighbourhood frames are easily seen to be coalgebras for the contravariant powerset functor composed with itself, denoted $\mathcal{2}^{\mathcal{2}}$. From a coalgebra point of view, neighbourhood structures are interesting since they constitute a general framework for studying coalgebraic modal logics in the style of Pattinson [111], where modalities are defined in terms of predicate liftings. It can easily be shown that any (unary) modality defined in this way, can be viewed as a neighbour-

hood modality. Furthermore, in much work on coalgebra (cf. [129]) it is often assumed that the functor preserves weak pullbacks, but it is not always clear whether this requirement is really needed. In [51], weaker functor requirements for congruences are studied, and $\mathcal{2}^{\mathcal{2}}$ provides an example of a functor which does not preserve weak pullbacks in general, but only the special ones consisting of kernel pairs.

From the modal logic point of view, coalgebra is interesting since it offers an abstract theory which can be instantiated to neighbourhood models, and help us generalise the well-known Kripke notions such as bisimilarity and image-finiteness to neighbourhood models. For monotonic neighbourhood structures, these questions have already been addressed (cf. [113, 52, 56]), but as mentioned in [113], if one starts from elementary intuitions, it is not immediately clear how to generalise monotonic bisimulation to arbitrary neighbourhood structures. The theory of coalgebra provides us not with one, but with several notions of state equivalence in F-coalgebras for an arbitrary functor F. F-bisimilarity and behavioural equivalence are well-known concepts, and it is generally known that the two notions coincide if and only if the functor F preserves weak pullbacks [129]. This is, for example, the case over Kripke frames which are coalgebras for the covariant powerset functor $\mathcal{P}$, and it explains some of the fundamental properties of Kripke bisimulation: (i) Kripke bisimulations are characterised by back-and-forth conditions, which makes it possible to efficiently compute Kripke bisimilarity over finite models as a greatest fixed point. (ii) The Hennessy-Milner theorem for normal modal logic states that over the class of finite Kripke models, two states are Kripke bisimilar if and only if they satisfy the same modal formulas. (iii) Van Benthem's characterisation theorem [19, 20] tells us that Kripke bisimilarity characterises the modal fragment of first-order logic. These properties of Kripke bisimulations form the starting points of our investigation into equivalence notions in neighbourhood structures and classical modal logic.

As neighbourhood structures are coalgebras for a functor that does not preserve weak pullbacks, it is to be expected that only behavioural equivalence will give rise to a Hennessy-Milner theorem for classical modal logic. However, it turns out to be very difficult to give a back-and-forth style characterisation of behavioural equivalence. This motivates our introduction of a third equivalence notion whose witnessing relations we call precocongruences. Precocongruences can be seen as a generalisation of the notion of a precongruence from [1].

The main contributions of this chapter are: (1) the introduction of precocongruences and basic results which relate them to bisimulations and behavioural equivalence. In particular, we show that on a single coalgebra, the largest precocongruence is behavioural equivalence (Theorem 5.3.11), and that over neighbourhood models, precocongruences are a better approximation of behavioural equivalence than $\mathcal{2}^{\mathcal{2}}$-bisimilarity; (2) the definition of a notion of modal saturation for neighbourhood models, which leads to a behavioural-equivalence-

somewhere-else result (Theorem 5.4.26) by showing that ultrafilter extensions are a Hennessy-Milner class; (3) a Van Benthem style characterisation of the classical modal fragment of first-order logic (Theorem 5.5.5); and (4) a model-theoretic proof of Craig interpolation for classical modal logic (Theorem 5.5.11).

In section 5.2 we define basic notions and notation. In section 5.3, we define precocongruences and investigate their relationship with bisimulations and behavioural equivalence. We also instantiate all three notions to the concrete case of neighbourhood frames, provide back-and-forth style characterisations for $\mathcal{2}^{\mathcal{2}}$-bisimulations and precocongruences, and prove the results mentioned in (1). In section 5.4, we introduce our notion of modal saturation for neighbourhood models, and use it to prove a Hennessy-Milner theorem for the class of finite neighbourhood models. We then use general coalgebraic constructions to define image-finite neighbourhood models and ultrafilter extensions of neighbourhood models, and show that these are also Hennessy-Milner classes. Finally, in section 5.5 we prove our main results as described in (3) and (4) above. In particular, we demonstrate that $\mathcal{2}^{\mathcal{2}}$-bisimulations are a useful tool for proving Craig interpolation of classical modal logic.

Since neighbourhood structures are of general interest outside the world of coalgebra, we have tried to keep the material of this chapter accessible to readers who are not familiar with coalgebraic modal logic. This means that some of our results could be obtained by instantiating more general results in coalgebra. When this is the case, we give a brief explanation in the form of a remark of how the general coalgebraic framework instantiates to neighbourhood structures. However, these remarks are not necessary for understanding the main results of the chapter. On the other hand, we also hope that these remarks will inspire readers to study the more general results.

5.2 Preliminaries and notation

In this section, we settle on notation, define the necessary set-theoretic notions, and introduce neighbourhood semantics for modal logic.

We assume the reader is familiar with the Kripke semantics and the basic model theory of normal modal logic. Some knowledge of more advanced topics such as modal saturation and ultrafilter extensions will be useful. All the necessary background information can be found in [25]. Extensive discussions on neighbourhood semantics can be found in [35, 40, 52, 141].

5.2.1 Functions and relations

Let X and Y be sets. If $Y \subseteq X$, then we write ι_Y for the inclusion map $\iota_Y : Y \hookrightarrow X$; Y^c for the complement $X \setminus Y$ of Y in X; and $Y \subseteq_\omega X$ if Y is a

finite subset of X. For a subset $Y \subseteq X$, we write $\uparrow Y = \{Y' \subseteq X \mid Y \subseteq Y'\}$ for the *upwards closure of* $\{Y\}$ *in* $\mathcal{P}(X)$.

For a function $f\colon X \to Y$ and subsets $U \subseteq X$ and $V \subseteq Y$ we define the *direct f-image of U* and the *f-preimage of V* by putting $f[U] := \{f(x) \mid x \in U\}$ and $f^{-1}[V] := \{x \in X \mid f(x) \in V\}$, respectively. Furthermore we call $\mathrm{dom}(f) := X$ the domain of f and we call $\mathrm{rng}(f) := f[X]$ the *range of f*. More generally, we also define the notions image, preimage, domain and range for a relation $R \subseteq X \times Y$. For $U \subseteq X$ and $V \subseteq Y$, we denote the *R-image of U* by $R[U] = \{y \in Y \mid \exists x \in U : xRy\}$, and the *$R$-preimage of V* by $R^{-1}[V] = \{x \in X \mid \exists y \in V : xRy\}$. The *domain of R* is $\mathrm{dom}(R) = R^{-1}[Y]$, and the *range of R* is $\mathrm{rng}(R) = R[X]$. We will often work with a relation in terms of its projection maps. Let $R \subseteq X_1 \times X_2$ be a relation. The maps $\pi_1 : R \to X_1$ and $\pi_2 : R \to X_2$ denote the *projections* defined for all $\langle x_1, x_2\rangle \in R$ by $\pi_i(\langle x_1, x_2\rangle) = x_i$, for $i = 1, 2$. R is called a *full* relation if π_1 and π_2 are surjective. Note that for $U_i \subseteq X_i$, $i = 1, 2$, we have $R[U_1] = \pi_2[\pi_1^{-1}[U_1]]$ and $R^{-1}[U_2] = \pi_1[\pi_2^{-1}[U_2]]$.

Recall from Chapter 2 that if $R \subseteq X \times X$, then we denote by R^e the smallest equivalence relation on X which contains R, and if R is an equivalence relation on X then X/R is the set of R-equivalence classes. A relation $R \subseteq X_1 \times X_2$, can be viewed as a relation $R_{X_1+X_2}$ on $X_1 + X_2$ by composing the projections with the canonical inclusion maps $\iota_1 : X_1 \to X_1 + X_2$ and $\iota_2 : X_2 \to X_1 + X_2$. More precisely, $R_{X_1+X_2} = \{\langle \iota_1(x_1), \iota_2(x_2)\rangle \mid \langle x_1, x_2\rangle \in R\}$.

Throughout this chapter the notion of coherence will be used extensively.

5.2.1. Definition. Let X_1 and X_2 be sets, $R \subseteq X_1 \times X_2$ a relation, $U_1 \subseteq X_1$ and $U_2 \subseteq X_2$. The pair $\langle U_1, U_2\rangle$ *is R-coherent* if: $R[U_1] \subseteq U_2$ and $R^{-1}[U_2] \subseteq U_1$. For a set X, a relation $R \subseteq X \times X$ and $U \subseteq X$, we say that U *is R-coherent*, if $\langle U, U\rangle$ is R-coherent. ◁

If $R \subseteq X_1 \times X_2$, then trivially $\langle \emptyset, \emptyset\rangle$ and $\langle X_1, X_2\rangle$ are R-coherent. Note that if R is an equivalence relation, then an R-coherent subset U is often called *R-closed*. We list a number of useful properties of R-coherence in the following two lemmas. Their easy, but instructive, proofs are left to the reader.

5.2.2. Lemma. *Let $R \subseteq X_1 \times X_2$ be a relation with projections $\pi_i : R \to X_i$, $i = 1, 2$. For all $U_1 \subseteq X_1$ and $U_2 \subseteq X_2$, the following are equivalent:*

1. *$\langle U_1, U_2\rangle$ is R-coherent.*
2. *for all $\langle x_1, x_2\rangle \in R$: $x_1 \in U_1 \Leftrightarrow x_2 \in U_2$.*
3. *$\pi_1^{-1}[U_1] = \pi_2^{-1}[U_2]$.*
4. *$U_1 + U_2$ is $R_{X_1+X_2}$-coherent.*

5.2.3. LEMMA. *Let $R \subseteq X \times X$ be a relation and $U \subseteq X$. The following are equivalent:*

1. *U is R-coherent.*
2. *U is R^e-coherent, i.e. R^e-closed.*
3. *U is a union of R^e-equivalence classes.*
4. *U^c is R^e-coherent.*

5.2.2 Classical modal logic and neighbourhood semantics

Let $\mathsf{At} = \{p_j \mid j \in \omega\}$ be a countable set of atomic sentences. The *basic modal language over* At, denoted $\mathcal{L}(\mathsf{At})$, is defined by the grammar:

$$\varphi ::= \bot \mid p_j \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box\varphi,$$

where $j \in \omega$. We define $\top$, $\rightarrow$ and $\leftrightarrow$ in the usual way. We will assume At to be fixed, and to ease notation, we write $\mathcal{L}$ instead of $\mathcal{L}(\mathsf{At})$.

5.2.4. DEFINITION. A *neighbourhood frame* is a pair $\langle S, \nu\rangle$ where S is a set of states and $\nu\colon S \to \mathcal{P}(\mathcal{P}(S))$ is a neighbourhood function which assigns to each state $s \in S$ its collection of neighbourhoods $\nu(s)$. A *neighbourhood model* based on a neighbourhood frame $\langle S, \nu\rangle$ is a triple $\langle S, \nu, V\rangle$ where $V\colon \mathsf{At} \to \mathcal{P}(S)$ is a valuation function. $\lhd$

Given a neighbourhood model $\mathcal{M}$, a state s in $\mathcal{M}$ and an $\mathcal{L}$-formula φ, we write $\mathcal{M}, s \models \varphi$ to denote that φ is true at s in $\mathcal{M}$, and $\mathcal{M}, s \not\models \varphi$, if φ is not true at s in $\mathcal{M}$. Truth of $\mathcal{L}$ in neighbourhood models is inductively defined as follows. Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model, $s \in S$ and $\varphi, \psi \in \mathcal{L}$.

$$\begin{array}{lll}
\mathcal{M}, s \not\models \bot & & \\
\mathcal{M}, s \models p_j & \text{iff} & s \in V(p_j) \quad \text{for } p_j \in \mathsf{At}, \\
\mathcal{M}, s \models \neg\varphi & \text{iff} & \mathcal{M}, s \not\models \varphi, \\
\mathcal{M}, s \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, s \models \varphi \text{ and } \mathcal{M}, s \models \psi, \\
\mathcal{M}, s \models \Box\varphi & \text{iff} & [\![\varphi]\!]^{\mathcal{M}} \in \nu(s),
\end{array} \tag{5.1}$$

where $[\![\varphi]\!]^{\mathcal{M}} = \{t \in S \mid \mathcal{M}, t \models \varphi\}$ denotes the *truth set of φ in $\mathcal{M}$*. Let also $\mathcal{N}$ be a neighbourhood model. Two states, s in $\mathcal{M}$ and t in $\mathcal{N}$, are *modally equivalent* (notation: $\mathcal{M}, s \equiv \mathcal{N}, t$ or simply $s \equiv t$), if they satisfy the same modal $\mathcal{L}$-formulas, i.e., $s \equiv t$ if and only if for all $\varphi \in \mathcal{L}$: $\mathcal{M}, s \models \varphi$ iff $\mathcal{N}, t \models \varphi$. A subset $X \subseteq S$ is *modally coherent*, if for all $s, t \in S$ such that $s \equiv t$: $s \in X$ iff $t \in X$ i.e., X is $\equiv$-coherent. A subset $X \subseteq S$ is *modally definable*, if there is a formula $\varphi \in \mathcal{L}$ such that $X = [\![\varphi]\!]^{\mathcal{M}}$.

Let $\Phi \cup \{\varphi\} \subseteq \mathcal{L}$. We write $\Phi \models \varphi$ if φ is a *local semantic consequence* of Φ over the class of all neighbourhood models, i.e., for any neighbourhood model $\mathcal{M}$ and state s in $\mathcal{M}$, if $\mathcal{M}, s \models \Phi$ then $\mathcal{M}, s \models \varphi$. In particular, if $\Phi \not\models \bot$ then Φ is called *consistent*, which means that Φ is satisfiable in some neighbourhood model, and $\models \varphi$ means that φ is valid in all neighbourhood models. We define *classical modal logic* $\mathbf{E}$ to be the theory of neighbourhood models, that is, for all $\mathcal{L}$-formulas φ: $\varphi \in \mathbf{E}$ iff $\models \varphi$. We will not be concerned with proof theory or axiomatics. For these matters, the reader is referred to [35].

The structure preserving maps between neighbourhood structures will be referred to as bounded morphisms. These have previously been studied in the context of algebraic duality [40], and monotonic neighbourhood structures [52] (we define monotonic neighbourhood structures below).

5.2.5. Definition. If $\mathcal{M}_1 = \langle S_1, \nu_1, V_1\rangle$ and $\mathcal{M}_2 = \langle S_2, \nu_2, V_2\rangle$ are neighbourhood models, and $f\colon S_1 \to S_2$ is a function, then f is a *(frame) bounded morphism from* $\langle S_1, \nu_1\rangle$ *to* $\langle S_2, \nu_2\rangle$ (notation: $f\colon \langle S_1, \nu_1\rangle \to \langle S_2, \nu_2\rangle$) if for all $s \in S_1$ and all $X \subseteq S_2$:

$$f^{-1}[X] \in \nu_1(s) \;\text{ iff }\; X \in \nu_2(f(s)). \tag{5.2}$$

If also $s \in V_1(p_j)$ iff $f(s) \in V_2(p_j)$, for all $p_j \in \mathsf{At}$, and all $s \in S_1$, then f is a *bounded morphism from* $\mathcal{M}_1$ *to* $\mathcal{M}_2$ (notation: $f\colon \mathcal{M}_1 \to \mathcal{M}_2$). ◁

Bounded morphisms preserve truth of modal formulas.

5.2.6. Lemma. *Let* $\mathcal{M}_1 = \langle S_1, \nu_1, V_1\rangle$ *and* $\mathcal{M}_2 = \langle S_2, \nu_2, V_2\rangle$ *be two neighbourhood models and* $f\colon \mathcal{M}_1 \to \mathcal{M}_2$ *a bounded morphism. For each modal formula* $\varphi \in \mathcal{L}$ *and state* $s \in S_1$, $\mathcal{M}_1, s \models \varphi$ *iff* $\mathcal{M}_2, f(s) \models \varphi$.

Proof. By a straightforward induction on the formula structure. Details left to the reader. QED

Neighbourhood frames and bounded (frame) morphisms form a category which we denote by NbhdFr. Similarly, neighbourhood models and bounded morphisms form a category Nbhd. This can easily be verified directly, but it also follows from the straightforward coalgebraic modelling of neighbourhood strcutures which we describe now.

Recall that $\mathcal{2}^{\mathcal{2}}$ denotes the Set-functor obtained by composing the contravariant powerset functor $\mathcal{2}$ with itself, i.e., $\mathcal{2}^{\mathcal{2}} = \mathcal{2} \circ \mathcal{2}$. More specifically, for any set X and any function $f\colon X \to Y$,

$$\begin{array}{lcl} \mathcal{2}^{\mathcal{2}}(X) & = & \mathcal{P}(\mathcal{P}(X)), \\ \mathcal{2}^{\mathcal{2}}(f)(U) & = & \{D \subseteq Y \mid f^{-1}[D] \in U\} \text{ for all } U \in \mathcal{2}^{\mathcal{2}}(X). \end{array}$$

5.2.7. Proposition. $\mathsf{NbhdFr} = \mathsf{Coalg}(\mathcal{2}^{\mathcal{2}})$ *and* $\mathsf{Nbhd} = \mathsf{Coalg}(\mathcal{2}^{\mathcal{2}}(-) \times \mathcal{P}(\mathsf{At}))$.

PROOF. A neighbourhood frame $\langle S, \nu \rangle$ is clearly an object in $\mathsf{Coalg}(\mathcal{2}^{\mathcal{2}})$. Given a neighbourhood model $\langle S, \nu, V \rangle$, we can view the valuation $V\colon \mathsf{At} \to \mathcal{P}(S)$ in its transposed form $\hat{V}\colon S \to \mathcal{P}(\mathsf{At})$ where $p_j \in \hat{V}(s)$ iff $s \in V(p_j)$. It is now easy to see that $\langle S, \nu, V \rangle$ uniquely corresponds to a coalgebra $\langle \nu, \hat{V} \rangle\colon S \to \mathcal{2}^{\mathcal{2}}(S) \times \mathcal{P}(\mathsf{At})$ for the functor $\mathcal{2}^{\mathcal{2}}(-) \times \mathcal{P}(\mathsf{At})$.

Now suppose $\langle S_1, \nu_1 \rangle$ and $\langle S_2, \nu_2 \rangle$ are $\mathcal{2}^{\mathcal{2}}$-coalgebras. A function $f\colon S_1 \to S_2$ is a $\mathcal{2}^{\mathcal{2}}$-coalgebra morphism iff for all $s \in S_1$: $\nu_2(f(s)) = \mathcal{2}^{\mathcal{2}}(f)(\nu_1(s))$, i.e., for all $U \in \mathcal{2}^{\mathcal{2}}(S_2)$: $U \in \nu_2(f(s))$ iff $U \in \mathcal{2}^{\mathcal{2}}(f)(\nu_1(s))$ which by definition of $\mathcal{2}^{\mathcal{2}}(f)$ is equivalent with, $U \in \nu_2(f(s))$ iff $f^{-1}[U] \in \nu_1(s)$. Hence $\mathcal{2}^{\mathcal{2}}$-coalgebra morphisms are exactly the bounded frame morphisms, and we have $\mathsf{NbhdFr} = \mathsf{Coalg}(\mathcal{2}^{\mathcal{2}})$. It is just as easily shown that the morphisms in $\mathsf{Coalg}(\mathcal{2}^{\mathcal{2}}(-) \times \mathcal{P}(\mathsf{At}))$ are exactly the bounded morphisms between neighbourhood models. Hence $\mathsf{Nbhd} = \mathsf{Coalg}(\mathcal{2}^{\mathcal{2}}(-) \times \mathcal{P}(\mathsf{At}))$. QED

From now on, we will switch freely between the coalgebraic setting and the neighbourhood setting.

In the course of this chapter, we will relate some of our results to existing results for monotonic modal logic and normal modal logic. We briefly remind the reader of their definitions and their relationship with neighbourhood structures and coalgebras.

5.2.8. REMARK. A neighbourhood frame/model is *monotonic*, if for all $s \in S$, the collection of neighbourhoods $\nu(s)$ is *upwards closed*, i.e., if $U \subseteq V$ and $U \in \nu(s)$ then $V \in \nu(s)$. Monotonic modal logic is the theory of monotonic neighbourhood models (cf. [35, 52]). It was shown in [56] that monotonic neighboourhood frames are coalgebras for the subfunctor Mon of $\mathcal{2}^{\mathcal{2}}$ which is defined by $Mon(X) = \{U \in \mathcal{P}(\mathcal{P}(X)) \mid U \text{ is upwards closed}\}$ on a set X. $\lhd$

5.2.9. REMARK. Kripke frames/models are in 1-1 correspondence with so-called augmented neighbourhood frames/models (cf. [35]). A neighbourhood frame $\langle S, \nu \rangle$ is *augmented*, if it is monotonic and for all $s \in S$, $\bigcap \nu(s) \in \nu(s)$. In other words, in an augmented neighbourhood frame, each neighbourhood collection is the upwards closure of a unique, smallest neighbourhood. Given a Kripke model $\mathcal{K} = \langle S, R, V \rangle$, we obtain an augmented neighbourhood model $\mathcal{K}^{\mathrm{aug}} = \langle S, \nu, V \rangle$, by taking $\nu(s) = {\uparrow} R[s]$ for all $s \in S$. Conversely, given an augmented neighbourhood model $\mathcal{M} = \langle S, \nu, V \rangle$, we define the Kripke model $\mathcal{M}^{\mathrm{krp}} = \langle S, R, V \rangle$ by taking $R[s] = \bigcap \nu(s)$ for all $s \in S$. Clearly, these transformations are inverses of each other. It is also straightforward to show that for any two Kripke models $\mathcal{K}_1$ and $\mathcal{K}_2$, a function is a Kripke bounded morphism from $\mathcal{K}_1$ to $\mathcal{K}_2$ iff f is a (neighbourhood) bounded morphism from $\mathcal{K}_1^{\mathrm{aug}}$ to $\mathcal{K}_2^{\mathrm{aug}}$. Hence the category of Kripke frames is isomorphic to the category of augmented neighbourhood frames. Moreover, a Kripke model $\mathcal{K}$ and its corresponding augmented model

$\mathcal{K}^{\mathrm{aug}}$ are pointwise equivalent, i.e., for all states s in $\mathcal{K}$ and any $\mathcal{L}$-formula φ: $\mathcal{K}, s \models \varphi$ iff $\mathcal{K}^{\mathrm{aug}}, s \models \varphi$. This can be proved by an easy induction on φ (cf. [35]). Normal modal logic is the logic of all Kripke models, or equivalently, of all augmented neighbourhood models.

We saw in Subsection 2.4.3 that Kripke frames and their bounded morphisms can be seen as the category of coalgebras and coalgebra morphisms for the covariant powerset functor $\mathcal{P}\colon \mathsf{Set} \to \mathsf{Set}$ which maps a set X to the powerset $\mathcal{P}(X)$, and a function $f\colon X \to Y$ to the direct image function $f[\_]\colon \mathcal{P}(X) \to \mathcal{P}(Y)$. ◁

5.2.3 Basic constructions

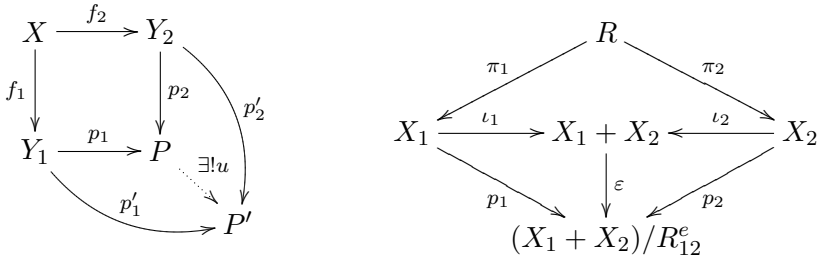
Finally, we will need a number of technical constructions. Disjoint unions of neighbourhood frames and models are just the coproducts in the relevant category. For neighbourhood models, this amounts to the following definition. The definition for neighbourhood frames is obtained by leaving out the part about the valuations.

5.2.10. Definition. Let $\mathcal{M}_1 = \langle S_1, \nu_1, V_1\rangle$ and $\mathcal{M}_2 = \langle S_2, \nu_2, V_2\rangle$ be two neighbourhood models, and let $\iota_i\colon S_i \to S_1 + S_2$, $i = 1, 2$, be the canonical inclusions. The *disjoint union of* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ is the neighbourhood model $\mathcal{M}_1 + \mathcal{M}_2 = \langle S_1 + S_2, \nu, V\rangle$ where for all $p \in \mathsf{At}$, $V(p) = \iota_1[V_1(p)] \cup \iota_2[V_2(p)]$; and for $i = 1, 2$, for all $X \subseteq S_1 + S_2$, and $s \in S_i$, $X \in \nu(s)$ iff $X \cap S_i \in \nu_i(s)$. ◁

Apart from coproducts and coequalisers, we will also make use of pushouts. We remind the reader of their definition. For more details, we refer to [3].

5.2.11. Definition. Let C be a category and let $f_1\colon X \to Y_1$ and $f_2\colon X \to Y_2$ be morphisms in C. A *pushout* of f_1 and f_2 in C is a triple $\langle P, p_1, p_1\rangle$ where P is an object and $p_1\colon Y_1 \to P$, $p_2\colon Y_2 \to P$ are morphisms in C such that $p_1 \circ f_1 = p_2 \circ f_2$. Moreover, if P', $p_1'\colon Y_1 \to P'$ and $p_2'\colon Y_2 \to P'$ are such that $p_1' \circ f_1 = p_2' \circ f_2$, then there exists a unique morphism $u\colon P \to P'$ in C such that $p_1' = u \circ p_1$ and $p_2' = u \circ p_2$, as illustrated in Figure 5.1(a). ◁

It is well-known that if the category C has coproducts and coequalisers, then a pushout of $f_1\colon X \to Y_1$ and $f_2\colon X \to Y_2$ in C can be constructed as a coequaliser of $\iota_1 \circ f_1, \iota_2 \circ f_2\colon X \to Y_1 + Y_2$, where $\iota_i\colon Y_i \to Y_1 + Y_2$, $i \in \{1, 2\}$, are the canonical inclusions into the coproduct. We are mainly interested in the case where f_1 and f_2 are the projection maps of some relation. In this case, the coproduct-coequaliser construction yields the following explicit construction in Set. Let $R \subseteq X_1 \times X_2$ be a relation with projections $\pi_1\colon R \to X_1$ and $\pi_2\colon R \to X_2$. Composing the projections with the canonical inclusions, we get the relation $R_{12} = R_{X_1+X_2} = \{\langle \iota_1(x_1), \iota_2(x_2)\rangle \mid \langle x_1, x_2\rangle \in R\}$ on $X_1 + X_2$. Let



(a) (b)

Figure 5.1: Pushout.

$\varepsilon\colon X_1 + X_2 \to (X_1 + X_2)/R^e_{12}$ be the associated quotient map. The *pushout of* R *in* Set is the triple $\langle P, p_1, p_2\rangle$, where $P := \big(X_1 + X_2\big)/R^e_{12}$, and $p_i = \varepsilon \circ \iota_i$, $i \in \{1, 2\}$. The construction is shown in Figure 5.1(b).

5.3 Equivalence notions

In this section we investigate three different equivalence notions over the class of neighbourhood frames, i.e., $\mathcal{2}^{\mathcal{2}}$-coalgebras. Two of these are behavioural equivalence and bisimilarity for the functor $\mathcal{2}^{\mathcal{2}}$. As $\mathcal{2}^{\mathcal{2}}$ is known not to preserve weak pullbacks, $\mathcal{2}^{\mathcal{2}}$-bisimilarity is strictly stronger than $\mathcal{2}^{\mathcal{2}}$-behavioural equivalence. This shortcoming of bisimilarity for non-weak pullback preserving functors motivates our introduction of a third equivalence notion whose witnessing relations we call precocongruences. Precocongruences generalise the idea behind precongruences (defined in [1]), which are relations on a single coalgebra, to relations between coalgebras.

In the first subsection, we work with an arbitrary functor F. First, we review the definition of precongruences and some results from [1]. Next, we define precocongruences and relate them to bisimulations, cocongruences and congruences. In the next subsection, we instantiate the three equivalence notions to neighbourhood frames, we give back-and-forth style characterisations of $\mathcal{2}^{\mathcal{2}}$-bisimulations and $\mathcal{2}^{\mathcal{2}}$-precocongruences, and we show by means of examples, that, even on finite neighbourhood frames, the three notions differ, but we also show that on a single neighbourhood frame, all three notions coincide. In the final subsection, we compare our equivalence notions with the existing notions of bisimulation for monotonic and Kripke frames.

5.3.1 Precocongruences

Throughout this subsection, we let $F\colon \mathsf{Set} \to \mathsf{Set}$ be an arbitrary functor. Precongruences were introduced by Aczel & Mendler in [1] as alternatives to bisimulations in the case the functor under consideration does not preserve weak pullbacks. We formulate the definition of precongruences in a slightly different way, and prove the equivalence with the definition from [1] in Lemma 5.3.2(1) below.

5.3.1. Definition (Precongruence). Let $\langle X, \xi\rangle$ be an F-coalgebra, $R \subseteq X \times X$ a relation, and $\varepsilon_R\colon X \to X/R^e$ the coequaliser of R. R is a *precongruence* on $\langle X, \xi\rangle$ if there exists a (necessarily unique) F-coalgebra structure λ on X/R^e such that ε_R is an F-coalgebra morphism from $\langle X, \xi\rangle$ to $\langle X/R^e, \lambda\rangle$ as illustrated by the following commuting diagram.

$$\begin{array}{ccccc} R & \overset{\pi_1}{\underset{\pi_2}{\rightrightarrows}} & X & \xrightarrow{\varepsilon_R} & X/R^e \\ & & \downarrow{\scriptstyle \xi} & & \vdots{\scriptstyle \exists!\lambda} \\ & & F(X) & \xrightarrow{F(\varepsilon_R)} & F(X/R^e) \end{array}$$

In other words, R is a precongruence on $\langle X, \xi\rangle$ iff R^e is a congruence on $\langle X, \xi\rangle$. ◁

The following lemma lists some useful facts on precongruences.

5.3.2. Lemma. *Let $\langle X, \xi\rangle$ be an F-coalgebra and $R \subseteq X \times X$ a relation.*

1. *R is a precongruence on $\langle X, \xi\rangle$ iff $F(\varepsilon_R) \circ \xi \circ \pi_1 = F(\varepsilon_R) \circ \xi \circ \pi_2$, i.e., $R \subseteq \ker(F(\varepsilon_R) \circ \xi)$.*
2. *The largest precongruence on $\langle X, \xi\rangle$ is the largest congruence on $\langle X, \xi\rangle$.*
3. *If R is a bisimulation then R is a precongruence.*

Proof. The proof can be found in [1], but we include it here for the sake of completeness.

For item 1, first note that the coequaliser of R is also the coequaliser of R^e. We now have: R is a precongruence iff R^e is a congruence iff (by Lemma 2.2.5, p. 14) $R^e \subseteq \ker(F(\varepsilon_R) \circ \xi)$. Since $R \subseteq R^e$ and kernels are equivalence relations, this last statement is equivalent with $R \subseteq \ker(F(\varepsilon_R) \circ \xi)$.

To see that item 2 holds, let S denote the union of all precongruences on $\langle X, \xi\rangle$. Note that if $R \subseteq S$, then $R^e \subseteq S^e$, and there exists a map $\varepsilon\colon X/R^e \to X/S^e$ such that $\varepsilon_S = \varepsilon \circ \varepsilon_R$. It follows that $F(\varepsilon_S) = F(\varepsilon) \circ F(\varepsilon_R)$, and hence

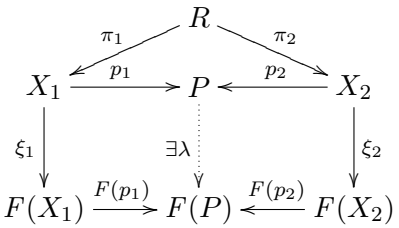
$$F(\varepsilon_S) \circ \xi = F(\varepsilon) \circ F(\varepsilon_R) \circ \xi. \tag{5.3}$$

Now if $\langle x_1, x_2\rangle \in S$, then by item 1, there is a relation $R \subseteq S$ such that $F(\varepsilon_R)(\xi(x_1)) = F(\varepsilon_R)(\xi(x_2))$. By (5.3), $F(\varepsilon_S)(\xi(x_1)) = F(\varepsilon_S)(\xi(x_2))$, which shows that S is a precongruence. S is necessarily also the largest congruence, since any congruence is a precongruence and therefore contained in S.

For item 3, suppose R is a bisimulation, i.e., the projection maps $\pi_1, \pi_2 \colon R \to X$ are F-coalgebra morphisms. In this case, the coequaliser ε_R of π_1 and π_2 is also the coequaliser of π_1 and π_2 in $\mathsf{Coalg}(F)$, hence ε_R is an F-coalgebra morphism. It follows that R is a precongruence. QED

We wish to consider relations between coalgebras rather than just a relation on a single coalgebra. The definition of a precocongruence is essentially obtained by replacing coequalisers by pushouts.

5.3.3. Definition (Precocongruence). Let $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ be F-coalgebras, and let $R \subseteq X_1 \times X_2$ be a relation with pushout $\langle P, p_1, p_2\rangle$. The relation R is called a *precocongruence* between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$, if there exists a coalgebra map $\lambda \colon P \to F(P)$ such that the pushout maps $p_1 \colon X_1 \to P$ and $p_2 \colon X_2 \to P$ are F-coalgebra morphisms, i.e., the following diagram commutes:



In other words, R is a precocongruence iff its pushout $\langle P, p_1, p_2\rangle$ is a cocongruence. If two states x_1 and x_2 are related by some precocongruence, we write $x_1 \sim_p x_2$. ◁

The following lemma gives us a two alternative characterisations of precocongruences.

5.3.4. Lemma. *Let $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ be F-coalgebras, and let $R \subseteq X_1 \times X_2$ be a relation with pushout $\langle P, p_1, p_2\rangle$. The following are equivalent:*

1. *R is a precocongruence between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$.*
2. *$F(p_1) \circ \xi_1 \circ \pi_1 = F(p_2) \circ \xi_2 \circ \pi_2$, i.e., $R \subseteq \mathrm{pb}(F(p_1) \circ \xi_1, F(p_2) \circ \xi_2)$.*
3. *$R_{X_1+X_2}$ is a precongruence on $\langle X_1, \xi_1\rangle + \langle X_2, \xi_2\rangle$.*

PROOF. (1 $\Leftrightarrow$ 2): Item 2 holds iff the outer part of the diagram in Definition 5.3.3 commutes, so the implication (1 $\Rightarrow$ 2) is immediate. Conversely, if item 2 holds, then by the universal property of the pushout $\langle P, p_1, p_2\rangle$ there is a (unique) function $\lambda\colon P \to F(P)$ such that $\lambda \circ p_1 = F(p_1) \circ \xi_1$ and $\lambda \circ p_2 = F(p_2) \circ \xi_2$. Hence R is a precocongruence,

(1 $\Rightarrow$ 3): If the pushout maps are morphisms, there exists by the universal property of the coproduct $\langle X_1, \xi_1\rangle + \langle X_2, \xi_2\rangle$ in $\mathsf{Coalg}(F)$, a unique F-coalgebra morphism $u\colon X_1 + X_2 \to P$ such that $p_i = u \circ \iota_i$, $i \in \{1, 2\}$. By the definition and construction of the pushout (cf. Figure 5.1(b)), it must be the case that u is equal to the natural quotient map $\varepsilon\colon X_1 + X_2 \to P$, and hence $R_{X_1+X_2}$ is a precongruence.

(3 $\Rightarrow$ 1): If $R_{12} = R_{X_1+X_2}$ is a precongruence on the coproduct, then the quotient map $\varepsilon\colon X_1 + X_2 \to (X_1 + X_2)/R^e_{12}$ is an F-coalgebra morphism. Since $p_i = \varepsilon \circ \iota_i$, $i \in \{1, 2\}$, and the canonical inclusions $\iota_i\colon X_i \to X_1 + X_2$, $i \in \{1, 2\}$, are also F-coalgebra morphisms, it follows that the pushout maps are F-coalgebra morphisms. QED

An interesting property of precocongruences, is that, like bisimulations, they can be characterised by a form of relation lifting.

5.3.5. DEFINITION. Let $R \subseteq X_1 \times X_2$ be a relation and let $\langle P, p_1, p_2\rangle$ be the pushout of $\langle R, \pi_1, \pi_2\rangle$. We define the F-lifting $\mathit{Lif}(F)(R) \subseteq F(X_1) \times F(X_2)$ of R by

$$\mathit{Lif}(F)(R) \quad := \quad \mathrm{pb}(F(p_1), F(p_2)). \qquad \lhd$$

Note that $\mathit{Lif}(F)$ is independent of the concrete representation of the pushout. This follows easily from the fact that pushouts are unique up-to isomorphism. The definition of $\mathit{Lif}(F)$ goes back to an idea by Kurz (personal comunication) for defining a relation lifting of functors that do not preserve weak pullbacks.

5.3.6. LEMMA. *Let $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ be F-coalgebras, and let $R \subseteq X_1 \times X_2$ be a relation. R is precocongruence iff*

$$\text{for all } \langle x_1, x_2\rangle \in R\text{:} \quad \langle \xi_1(x_1), \xi_2(x_2)\rangle \in \mathit{Lif}(F)(R).$$

PROOF. Immediate from Lemma 5.3.4 and the definition of $\mathit{Lif}(F)$. QED

The characterisation of precocongruences in Lemma 5.3.6 makes it easy to show that between any two coalgebras, there exists a largest, and necessarily unique, precocongruence. First, note that for any relations $R' \subseteq R \subseteq X_1 \times X_2$ with pushouts $\langle P', p'_1, p'_2\rangle$ and $\langle P, p_1, p_2\rangle$, respectively, there exists by the universal property of P' a unique map $u\colon P' \to P$ such that $p_i = u \circ p'_i$, $i \in \{1, 2\}$. Consequently, $F(p_i) = F(u) \circ F(p'_i)$, $i \in \{1, 2\}$, and for all $t_1 \in F(X_1)$, $t_2 \in F(X_2)$: $F(p'_1)(t_1) = F(p'_2)(t_2)$ implies that $F(p_1)(t_1) = F(p_2)(t_2)$. Hence,

$$R' \subseteq R \quad \Rightarrow \quad \mathit{Lif}(F)(R') \subseteq \mathit{Lif}(F)(R). \tag{5.4}$$

5.3.7. Lemma. *Let $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ be F-coalgebras. The union of all precocongruences between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ is again a precocongruence.*

Proof. Let R be the union of all precocongruences between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$, let $\langle P, p_1, p_2\rangle$ be the pushout of R, and assume $\langle x_1, x_2\rangle \in R$. Then there is a precocongruence $R' \subseteq R$ such that $\langle x_1, x_2\rangle \in R'$. Letting $\langle P', p_1', p_2'\rangle$ be the pushout of R', it follows that $\langle \xi_1(x_1), \xi_2(x_2)\rangle \in \mathit{Lif}(F)(R')$, and hence by (5.4) that $\langle \xi_1(x_1), \xi_2(x_2)\rangle \in \mathit{Lif}(F)(R)$. We conclude by Lemma 5.3.6 that R is a precocongruence. QED

In the following proposition we give a first comparison between precocongruences, bisimulations and cocongruences.

5.3.8. Proposition. *Let $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ be F-coalgebras, and let R be a relation between X_1 and X_2.*

1. *If R is a bisimulation, then R is a precocongruence.*
2. *If R is a precocongruence, then R is contained in a cocongruence.*

Consequently, for all $x_1 \in X_1$ and $x_2 \in X_2$:

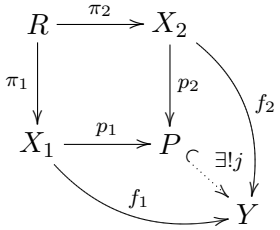
$$x_1 \sim x_2 \;\; \textit{implies} \;\; x_1 \sim_p x_2 \;\; \textit{implies} \;\; x_1 \sim_b x_2.$$

Proof. Let $R \subseteq X_1 \times X_2$ be a relation with projections $\pi_1 \colon R \to X_1$ and $\pi_2 \colon R \to X_2$, and pushout $\langle P, p_1, p_2\rangle$. Item 1: Assume R is a bisimulation. By composing the projections with the canonical inclusion morphisms into the coproduct, we have a pair of parallel F-coalgebra morphisms $\iota_1 \circ \pi_1, \iota_2 \circ \pi_2 \colon R \to X_1 + X_2$. The quotient map $\varepsilon \colon X_1 + X_2 \to (X_1 + X_2)/R^e_{X_1+X_2}$ is now the coequaliser of $\iota_1 \circ \pi_1$ and $\iota_2 \circ \pi_2$ also in $\mathsf{Coalg}(F)$, hence an F-coalgebra morphism. It follows that p_1 and p_1 are also F-coalgebra morphisms, since $p_i = \varepsilon \circ \iota_i$, $i \in \{1, 2\}$. Item 2: If R is a precocongruence, then the pushout maps p_1 and p_2 are F-coalgebra morphisms. The claim now follows from the fact that $R \subseteq \mathrm{pb}(p_1, p_2)$. QED

Proposition 5.3.8 alone does not yet tell us whether precocongruences are a better approximation of behavioural equivalence than F-bisimulations, but in the next subsection, we will see that, in general, the implications of Proposition 5.3.8 are strict. The following lemma provides us with a criterion which ensures that a cocongruence is a precocongruence.

5.3.9. Lemma. *If $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ are F-coalgebras and $R \subseteq X_1 \times X_2$ is a full cocongruence between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$, then R is a precocongruence.*

PROOF. Let R be a cocongruence with projection maps $\pi_1 \colon R \to X_1$ and $\pi_2 \colon R \to X_2$ and pushout $\langle P, p_1, p_2 \rangle$. Then there exist an F-coalgebra $\langle Y, \gamma \rangle$ and F-coalgebra morphisms $f_i \colon X_i \to Y$ for $i \in \{1, 2\}$ such that $R = \mathrm{pb}(f_1, f_2)$. We are going to define a function $\lambda \colon P \to F(P)$ such that p_i is an F-coalgebra morphism from $\langle X_i, \xi_i \rangle$ to $\langle P, \lambda \rangle$ for $i \in \{1, 2\}$. By the universal property of the pushout there has to be a function $j \colon P \to Y$ such that $j \circ p_i = f_i$ for $i \in \{1, 2\}$, (see diagram). We claim that the map j is injective. First, it follows from the definition of the pushout that both p_1 and p_2 are surjective, because R is full. Let now $z_1, z_2 \in P$ and suppose that $j(z_1) = j(z_2)$. The surjectivity of the p_i's implies that there are $s_1 \in X_1$ and $s_2 \in X_2$ such that $p_1(s_1) = z_1$ and $p_2(s_2) = z_2$. Hence $j(p_1(s_1)) = j(p_2(s_2))$, i.e., $f_1(s_1) = f_2(s_2)$, which implies that $\langle s_1, s_2 \rangle \in R$ and consequently we get $p_1(s_1) = p_2(s_2)$, i.e., $z_1 = z_2$. This demonstrates that j is injective and thus there is some surjective map $e \colon Y \to P$ with $e \circ j = id_P$. Now define $\lambda := F(e) \circ \lambda \circ j$. It remains to check that for $i \in \{1, 2\}$, $p_i \colon \langle X_i, \xi_i \rangle \to \langle P, \lambda \rangle$ is an F-coalgebra morphism. Let $i \in \{1, 2\}$, we have:



$$\begin{array}{rcll} \lambda \circ p_i & = & F(e) \circ \lambda \circ j \circ p_i & \\ & = & F(e) \circ \lambda \circ f_i & (j \circ p_i = f_i) \\ & = & F(e) \circ F(f_i) \circ \xi_i & (f_i \text{ coalg. morph.}) \\ & = & F(e) \circ F(j) \circ F(p_i) \circ \xi_i & (F(f_i) = F(j) \circ F(p_i)) \\ & = & F(p_i) \circ \xi_i & (F(e) \circ F(j) = id_{F(P)}) \end{array}$$ QED

We introduced precocongruences as a generalisation of precongruences to relations between different coalgebras. We point out that this generalisation is conceptual rather than set-theoretic, since on a single coalgebra, a precongruence is not necessarily a precocongruence (as we will see in Example 5.3.16 below). In fact, in the one-coalgebra case precocongruences specialise precongruences. This is the content of the first item of the next proposition. The second item tells us that over equivalence relations, (pre)congruences and precocongruences are the same.

5.3.10. PROPOSITION. *Let $\langle X, \xi \rangle$ be an F-coalgebra.*

1. *If $R \subseteq X \times X$ is a precocongruence on $\langle X, \xi \rangle$, then R is also a precongruence on $\langle X, \xi \rangle$.*
2. *If $R \subseteq X \times X$ is an equivalence relation then: R is a precocongruence on $\langle X, \xi \rangle$ iff R is a congruence on $\langle X, \xi \rangle$.*

PROOF. To prove item 1, let $\langle P, p_1, p_2 \rangle$ be the pushout of R, and let $\varepsilon_R \colon X \to X/R^e$ be the natural quotient map (i.e., the coequaliser of R). By the universal

property of the pushout in Set, there is a unique map $u\colon P \to X/R^e$ such that $u \circ p_1 = \varepsilon_R = u \circ p_2$. It follows that $F(u) \circ F(p_1) = F(\varepsilon_R) = F(u) \circ F(p_2)$, and hence for all $x, y \in X$: if $F(p_1)(\xi(x)) = F(p_2)(\xi(y))$ then $F(\varepsilon_R)(\xi(x)) = F(\varepsilon_R)(\xi(y))$. Now assume that R is a precocongruence on $\langle X, \xi\rangle$, i.e., between $\langle X, \xi\rangle$ and itself. By Lemma 5.3.4, for all $\langle x, y\rangle \in R$ we have that $F(p_1)(\xi(x)) = F(p_2)(\xi(y))$, and hence that $F(\varepsilon_R)(\xi(x)) = F(\varepsilon_R)(\xi(y))$. This shows that $R \subseteq \ker(F(\varepsilon_R) \circ \xi)$, and it now follows from Lemma 5.3.2(1) that R is a precongruence.

To prove item 2, we first note that if R is an equivalence relation and a precocongruence on $\langle X, \xi\rangle$, then by item 1, R is also a precongruence and hence a congruence. Conversely, if R is a congruence, then R is clearly a full cocongruence on $\langle X, \xi\rangle$ and so by Lemma 5.3.9, a precocongruence. QED

From Proposition 5.3.10 it follows more or less immediately that behavioural equivalence is a precocongruence.

5.3.11. THEOREM. *If $\langle X, \xi\rangle$ is an F-coalgebra and $x_1, x_2 \in X$, then*

$$x_1 \sim_b x_2 \quad \textit{iff} \quad x_1 \sim_p x_2.$$

PROOF. Follows from Proposition 5.3.10(2) and the fact that behavioural equivalence is the largest congruence on $\langle X, \xi\rangle$ (cf. Lemma 2.3.3, p. 17). QED

Table 5.1 provides an informal overview of the congruence-like relations we have studied so far. The empty space in the cocongruence box reflects the observation that it seems impossible to check the existential requirement of a pair of coalgebra morphisms for which the relation is a pullback, in terms of some maps constructed from the relation itself, as in the case of congruences, precongruences and precocongruences.

R is a precongruence (R is contained in congruence) (coequaliser is a morphism)	R is a congruence (R is a kernel of a morphism) (coequaliser is a morphism)
R is a precocongruence (R is contained in cocongruence) (pushout maps are morphisms)	R is a cocongruence (R is a pullback of two morphisms)

Table 5.1: Informal overview of congruence-like relations.

Finally, Table 5.2 summarises the defining properties of the different types of relations that approximate behavioural equivalence in coalgebras.

relation	*coalgebra morphisms*	*coalgebra structure*
R is a bisimulation	projections π_1 and π_2	R
R is a precongruence	coequaliser $\varepsilon_R \colon X \to X/R^e$	X/R^e
R is a congruence	coequaliser $\varepsilon_R \colon X \to X/R$	X/R
R is a precocongruence	pushout maps p_1, p_2	pushout P
R is a cocongruence	some $f_i \colon X_i \to Y$, $i \in \{1,2\}$	some Y

Table 5.2: Defining properties of relations.

5.3.2 Equivalences between neighbourhood frames

In this subsection, we will investigate behavioural equivalence, bisimilarity and the equivalence notion arising from precocongruences over $\mathcal{2}^{\mathcal{2}}$-coalgebras, i.e., neighbourhood frames. First, we use the relation lifting characterisations of $\mathcal{2}^{\mathcal{2}}$-bisimulations and $\mathcal{2}^{\mathcal{2}}$-precocongruences to obtain set-theoretic, back-and-forth style predicates for these two equivalence notions. Next we provide examples which show that the implications from Proposition 5.3.8 are strict. However, we also show that on a single neighbourhood frame all three equivalence notions coincide. Finally, we compare the three equivalence notions with bisimulations over monotonic neighbourhood frames and Kripke frames.

5.3.12. Remark. For simplicity of presentation, we have chosen to only treat equivalence notions on neighbourhood frames, but the results of this section can easily be extended to neighbourhood *models*, i.e., $\mathcal{2}^{\mathcal{2}}(-) \times \mathcal{P}(\mathsf{At})$-coalgebras. For example, working out the details of the definition of $\mathcal{2}^{\mathcal{2}}(-) \times \mathcal{P}(\mathsf{At})$-bisimulation results in the expected characterisation: A relation R is $\mathcal{2}^{\mathcal{2}}(-) \times \mathcal{P}(\mathsf{At})$-bisimulation and if and only if R is a $\mathcal{2}^{\mathcal{2}}$-bisimulation and for all $\langle s, t\rangle \in R$, s and t satisfy the same atomic propositions. Similar statements hold for cocongruences and precocongruences. ◁

Let us start out by considering $\mathcal{2}^{\mathcal{2}}$-bisimulations. Let $\langle S_1, \nu_1\rangle$ and $\langle S_2, \nu_2\rangle$ be two $\mathcal{2}^{\mathcal{2}}$-coalgebras and $R \subseteq S_1 \times S_2$ a relation with projections $\pi_i \colon R \to S_i$, $i \in \{1,2\}$. Instantiating the relation lifting characterisation in terms of $Rel(F)$ from (2.1) and (2.2) (p. 18) for $F = \mathcal{2}^{\mathcal{2}}$, we find that:

$$
\begin{aligned}
& R \text{ is a } \mathcal{2}^{\mathcal{2}}\text{-bisimulation} \\
\text{iff} \quad & \forall \langle s_1, s_2\rangle \in R : \langle \nu_1(s_1), \nu_2(s_2)\rangle \in Rel(\mathcal{2}^{\mathcal{2}})(R) \\
\text{iff} \quad & \forall \langle s_1, s_2\rangle \in R : \langle \nu_1(s_1), \nu_2(s_2)\rangle \in \langle \mathcal{2}^{\mathcal{2}}(\pi_1), \mathcal{2}^{\mathcal{2}}(\pi_2)\rangle[\mathcal{2}^{\mathcal{2}}(R)] \\
\text{iff} \quad & \forall \langle s_1, s_2\rangle \in R \,.\, \exists Y \in \mathcal{2}^{\mathcal{2}}(R) : \nu_1(s_1) = \mathcal{2}^{\mathcal{2}}(\pi_1)(Y) \text{ and } \nu_2(s_2) = \mathcal{2}^{\mathcal{2}}(\pi_2)(Y) \\
\text{iff} \quad & \forall \langle s_1, s_2\rangle \in R \,.\, \exists Y \in \mathcal{2}^{\mathcal{2}}(R) \,.\, \forall i \in \{1,2\} \,.\, \forall U \subseteq S_i : \\
& \qquad\qquad U \in \nu_i(s_i) \Leftrightarrow \pi_i^{-1}[U] \in Y
\end{aligned}
\tag{5.5}
$$

Using the notion of R-coherence (cf. Lemma 5.2.2) we can reformulate the above requirement into a condition which does not involve any explicit mention of Y.

5.3.13. PROPOSITION. *Let $\mathcal{S}_1 = \langle S_1, \nu_1 \rangle$ and $\mathcal{S}_2 = \langle S_2, \nu_2 \rangle$ be neighbourhood frames. A relation $R \subseteq S_1 \times S_2$ is a $\mathcal{2}^{\mathcal{2}}$-*bisimulation *between $\mathcal{S}_1$ and $\mathcal{S}_2$ iff for all $\langle s_1, s_2 \rangle \in R$, for all $U_1, U_1' \subseteq S_1$ and for all $U_2, U_2' \subseteq S_2$ the following two conditions are satisfied:*

1.(a) If $\mathrm{dom}(R) \cap U_1 = \mathrm{dom}(R) \cap U_1'$ then: $U_1 \in \nu_1(s_1)$ iff $U_1' \in \nu_1(s_1)$, and

(b) If $\mathrm{rng}(R) \cap U_2 = \mathrm{rng}(R) \cap U_2'$ then: $U_2 \in \nu_2(s_2)$ iff $U_2' \in \nu_2(s_2)$.

2. If the pair $\langle U_1, U_2 \rangle$ is R-coherent then: $U_1 \in \nu_1(s_1)$ iff $U_2 \in \nu_2(s_2)$.

PROOF. First observe that for $U_1, U_1' \subseteq S_1$, $\pi_1^{-1}[U_1] = \pi_1^{-1}[U_1']$ iff $\mathrm{dom}(R) \cap U_1 = \mathrm{dom}(R) \cap U_1'$. Similarly, for $U_2, U_2' \subseteq S_2$, $\pi_2^{-1}[U_2] = \pi_2^{-1}[U_2']$ iff $\mathrm{rng}(R) \cap U_2 = \mathrm{rng}(R) \cap U_2'$. Also, recall (from Lemma 5.2.2) that given $U_1 \subseteq S_1$ and $U_2 \subseteq S_2$, the pair $\langle U_1, U_2 \rangle$ is R-coherent iff $\pi_1^{-1}[U_1] = \pi_2^{-1}[U_2]$. It should now be clear that if R is a $\mathcal{2}^{\mathcal{2}}$-bisimulation, i.e., (5.5) holds, then R fulfills conditions 1 and 2.

To prove the converse, assume R fulfills the conditions 1 and 2. To see that (5.5) holds, we can take for each pair $\langle s_1, s_2 \rangle \in R$ as witnessing $Y \in \mathcal{2}^{\mathcal{2}}(R)$, the set $Y := \{\pi_1^{-1}[U_1] \mid U_1 \in \nu_1(s_1)\} \cup \{\pi_2^{-1}[U_2] \mid U_2 \in \nu_2(s_2)\}$. To verify the condition on ν_1 in (5.5), we have that $U_1 \in \nu_1(s_1)$ implies $\pi_1^{-1}[U_1] \in Y$ simply by definition of Y. Now suppose $\pi_1^{-1}[U_1] \in Y$. This means that there is a $U_1' \in \nu_1(s_1)$ such that $\pi_1^{-1}[U_1] = \pi_1^{-1}[U_1']$ or there is a $U_2 \in \nu_2(s_2)$ such that $\pi_1^{-1}[U_1] = \pi_2^{-1}[U_2]$. In the first case, $U_1 \in \nu_1(s_1)$ follows from condition 1a, in the second case, $U_1 \in \nu_1(s_1)$ follows from condition 2. Similarly, the condition on ν_2 in (5.5) can be shown to hold using conditions 1b and 2. QED

Another way of formulating condition 1.(a) in Proposition 5.3.13, is to say that if $U_1 \in \nu_1(s_1)$ and $U_1' \notin \nu_1(s_1)$, then there is a $u \in (U_1 \setminus U_1') \cup (U_1' \setminus U_1)$ such that $u \in \mathrm{dom}(R)$. Similarly for condition 1b. Informally, one can say that condition 1 requires that the relation R must witness the difference between subsets when one is a neighbourhood and the other is not. We will now show that precocongruences are characterised by condition 2 only, hence condition 1 is unnecessary (unwanted even) for the purpose of approximating behavioural equivalence.

Let $\langle S_1, \nu_1 \rangle$ and $\langle S_2, \nu_2 \rangle$ be two $\mathcal{2}^{\mathcal{2}}$-coalgebras and $R \subseteq S_1 \times S_2$ a relation with pushout $\langle P, p_1, p_2 \rangle$. We have:

$$\begin{aligned} & R \text{ is a precocongruence} \\ \text{iff} \quad & \forall \langle s_1, s_2 \rangle \in R : \mathcal{2}^{\mathcal{2}}(p_1)(\nu_1(s_1)) = \mathcal{2}^{\mathcal{2}}(p_2)(\nu_2(s_2)) \\ \text{iff} \quad & \forall \langle s_1, s_2 \rangle \in R \,.\, \forall V \subseteq P : p_1^{-1}[V] \in \nu_1(s_1) \;\Leftrightarrow\; p_2^{-1}[V] \in \nu_2(s_2) \end{aligned} \tag{5.6}$$

We now show that, in fact, (5.6) is equivalent with condition 2 of Proposition 5.3.13.

5.3.14. Proposition. *Let $\mathcal{S}_1 = \langle S_1, \nu_1\rangle$ and $\mathcal{S}_2 = \langle S_2, \nu_2\rangle$ be neighbourhood frames, and $R \subseteq S_1 \times S_2$ a relation. R is a precocongruence between $\mathcal{S}_1$ and $\mathcal{S}_2$ if and only if for all $\langle s_1, s_2\rangle \in R$ and for all $U_1 \subseteq S_1$ and $U_2 \subseteq S_2$ such that $\langle U_1, U_2\rangle$ is R-coherent: $U_1 \in \nu_1(s_1)$ iff $U_2 \in \nu_2(s_2)$.*

Proof. Let $\mathcal{S}_1, \mathcal{S}_2$ and R be as stated. Furthermore, let $\pi_i \colon R \to S_i$, $i \in \{1,2\}$, be the projections of R, $R_{12} = R_{S_1+S_2}$, and $\langle P, p_1, p_2\rangle$ the pushout of R. We will prove that for all $U_1 \subseteq S_1$ and $U_2 \subseteq S_2$:

$$\langle U_1, U_2\rangle \text{ is } R\text{-coherent} \quad \text{iff} \quad U_1 = p_1^{-1}[Y] \text{ and } U_2 = p_2^{-1}[Y] \text{ for some } Y \subseteq P. \tag{5.7}$$

The proposition then follows from (5.6) and (5.7). To prove the direction from left to right in (5.7), assume $U_1 \subseteq S_2$, $U_2 \subseteq S_2$ and $\langle U_1, U_2\rangle$ is R-coherent. From Lemmas 5.2.2 and 5.2.3, we get that $U_1 + U_2$ is R_{12}^e-coherent. Let $\varepsilon \colon S_1 + S_2 \to P$ be the quotient map associated with R_{12}^e. We claim that we can take $Y = \varepsilon[U_1 + U_2]$, the set of R_{12}^e-equivalence classes intersecting $U_1 + U_2$. To see that $p_1^{-1}[\varepsilon[U_1 + U_2]] = U_1$ and $p_2^{-1}[\varepsilon[U_1 + U_2]] = U_2$, we have for all $i \in \{1,2\}$ and $s_i \in S_i$:

$$\begin{aligned} s_i \in p_i^{-1}[\varepsilon[U_1 + U_2]] &\iff p_i(s_i) \in \varepsilon[U_1 + U_2] \\ &\iff \exists s' \in U_1 + U_2 : \langle s_i, s'\rangle \in R_{12}^e \\ (U_1 + U_2 \ R_{12}^e\text{-coh.}) &\iff s_i \in U_1 + U_2 \\ &\iff s_i \in U_i. \end{aligned}$$

To prove the direction from right to left in (5.7), let $Y \subseteq P$ be arbitrary. We have for all $\langle s_1, s_2\rangle \in R$:

$$\langle s_1, s_2\rangle \in \pi_1^{-1}[p_1^{-1}[Y]] \text{ iff } p_1(s_1) \in Y \text{ iff } p_2(s_2) \in Y \text{ iff } \langle s_1, s_2\rangle \in \pi_2^{-1}[p_2^{-1}[Y]].$$

where the middle equivalence follows from the fact that $\langle s_1, s_2\rangle \in R$ implies $p_1(s_1) = p_2(s_2)$. We have now shown that $\pi_1^{-1}[p_1^{-1}[Y]] = \pi_2^{-1}[p_2^{-1}[Y]]$, hence by Lemma 5.2.2, the pair $\langle p_1^{-1}[Y], p_2^{-1}[Y]\rangle$ is R-coherent. QED

From Proposition 5.3.10(2) we know that on a single coalgebra, congruences are precocongruences. We therefore get the following characterisation.

5.3.15. Corollary. *Let $\langle S, \nu\rangle$ be a neighbourhood frame and $R \subseteq S \times S$ an equivalence relation. R is a congruence on $\langle S, \nu\rangle$ iff*

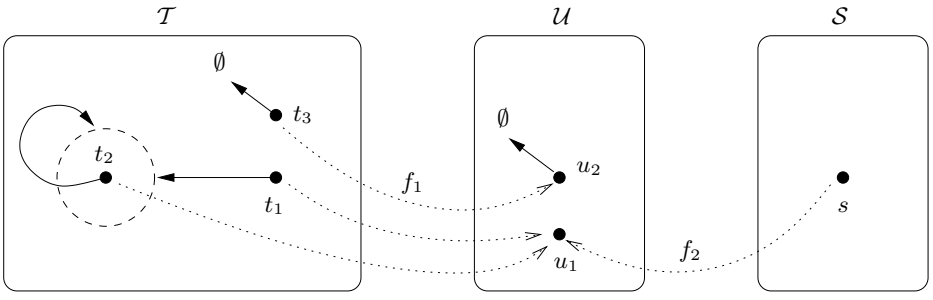
$$\text{for all } \langle s_1, s_2\rangle \in R \text{ and all } R\text{-coherent } U \subseteq S\text{: } U \in \nu(s_1) \text{ iff } U \in \nu(s_2). \tag{5.8}$$

PROOF. Let $R \subseteq S \times S$ be an equivalence relation. We first prove a small claim: *Claim:* A pair $\langle U_1, U_2 \rangle$ is R-coherent iff $U_1 = U_2 = U$ for some R-coherent subset $U \subseteq S$. *Proof of Claim:* Recall that a pair $\langle U_1, U_2 \rangle$ is R-coherent iff $R[U_1] \subseteq U_2$ and $R^{-1}[U_2] \subseteq U_1$. Since R is an equivalence relation, R is reflexive, and it follows that if $\langle U_1, U_2 \rangle$ is R-coherent, then $U_1 \subseteq R[U_1] \subseteq U_2$ and $U_2 \subseteq R^{-1}[U_2] \subseteq U_1$, hence $U_1 = U_2$. Conversely, if U is some R-coherent subset of S, then by definition, $\langle U, U \rangle$ is R-coherent.

We now have: R is a congruence iff (Thm. 5.3.11) R is a precocongruence iff (Prop. 5.3.14) for all $\langle s_1, s_2 \rangle \in R$ and for all $U_1, U_2 \subseteq S$ such that $\langle U_1, U_2 \rangle$ is R-coherent: $U_1 \in \nu(s_1)$ iff $U_2 \in \nu(s_2)$. Using the above claim, this last statement is equivalent with (5.8). QED

We will now demonstrate with two examples that $\mathcal{2}^{\mathcal{2}}$-bisimilarity, precocongruences and behavioural equivalence differ on neighbourhood frames. It is tempting to think of the elements of neighbourhoods as successor states, but these examples show that this leads to wrong intuitions. For example, contrary to the intuition we have from Kripke bisimulations, behavioural equivalence in neighbourhood frames does not require that nonempty neighbourhoods are somehow matched by nonempty neighbourhoods. Moreover, states that are not contained in any neighbourhood of some state s, can influence the existence of a bisimulation or cocongruence at s.

5.3.16. EXAMPLE. Consider the two neighbourhood frames, $\mathcal{T} = \langle T, \nu_T \rangle$ and $\mathcal{S} = \langle S, \nu_S \rangle$ where $T = \{t_1, t_2, t_3\}$, $\nu_T(t_1) = \nu_T(t_2) = \{\{t_2\}\}$, $\nu_T(t_3) := \{\emptyset\}$, and $S = \{s\}$, $\nu_S(s) = \emptyset$. The two states t_1 and s are behaviourally equivalent. To see this, let $\mathcal{U} = \langle U, \nu_U \rangle$ be the neighbourhood frame where $U = \{u_1, u_2\}$, $\nu_U(u_1) = \emptyset$ and $\nu_U(u_2) = \{\emptyset\}$. Let $f_1 \colon T \to U$ and $f_2 \colon S \to U$ be the functions with graphs $\mathrm{Gr}(f_1) = \{\langle t_1, u_1 \rangle, \langle t_2, u_1 \rangle, \langle t_3, u_2 \rangle\}$ and $\mathrm{Gr}(f_2) = \{\langle s, u_1 \rangle\}$, respectively, as illustrated here:



It can easily be verified that f_1 and f_2 are bounded morphisms. For example, the bounded morphism condition (5.2) holds for f_1 at t_1 and t_2, since their only neighbourhood $\{t_2\}$ is not the inverse f_1-image of any subset of U. Since $f_1(t_1) = f_2(s)$, t_1 and s are behaviourally equivalent. In fact, $R := \mathrm{pb}(f_1, f_2) =$

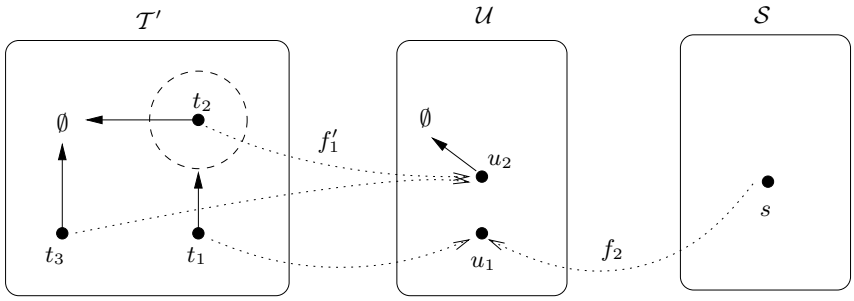
$\{\langle t_1, s\rangle, \langle t_2, s\rangle\}$ is a precocongruence. This can be verified using the characterisation given in Proposition 5.3.14. Note that there is no subset $U \subseteq S$ such that $\langle \{t_2\}, U\rangle$ is R-coherent.

However, t_1 and s are not $\mathcal{2}^{\mathcal{2}}$-bisimilar. For suppose R is a $\mathcal{2}^{\mathcal{2}}$-bisimulation between $\mathcal{T}$ and $\mathcal{S}$, then $\langle t_3, s\rangle \notin R$, since $\langle \emptyset, \emptyset\rangle$ is R-coherent, $\emptyset \in \nu_T(t_3)$ and $\emptyset \notin \nu_S(s)$. Hence $t_3 \notin \mathrm{dom}(R)$, and it follows that $\mathrm{dom}(R) \cap \{t_2\} = \mathrm{dom}(R) \cap \{t_2, t_3\}$. Now, since $\{t_2\} \in \nu_T(t_1)$ and $\{t_2, t_3\} \notin \nu_T(t_1)$, we can conclude from condition 1a of Proposition 5.3.13 that t_1 cannot be R-related to any state in $\mathcal{S}$, in particular not to s. Since R was an arbitrary $\mathcal{2}^{\mathcal{2}}$-bisimulation, t_1 and s are not $\mathcal{2}^{\mathcal{2}}$-bisimilar.

Consider now the relation $R' = \{\langle t_1, t_2\rangle\}$ on the neighbourhood frame $\mathcal{T}$. The reader can check that R' is a precongruence, but not a precocongruence, on $\mathcal{T}$. ◁

The above example shows that between neighbourhood frames, precocongruences are a better approximation of behavioural equivalence than $\mathcal{2}^{\mathcal{2}}$-bisimilarity. However, the next example shows that also precocongruences cannot capture behavioural equivalence, in general.

5.3.17. Example. We consider now a small variation on the picture given in Example 5.3.16. The neighbourhood frames $\mathcal{S}$, $\mathcal{U}$ and the function f_2 are the same as before, but on T we now take as neighbourhood function $\nu'_T(t_1) = \{\{t_2\}\}$, $\nu'_T(t_2) = \nu'_T(t_3) = \{\emptyset\}$, and let $\mathcal{T}' = \langle T, \nu'_T\rangle$. Instead of the function f_1, we take the function $f'_1 \colon T \to U$ with graph $\mathrm{Gr}(f'_1) = \{\langle t_1, u_1\rangle, \langle t_2, u_2\rangle, \langle t_3, u_2\rangle\}$. Again, it is straightforward to check that f'_1 is a bounded morphism, and hence t_1 and s are behaviourally equivalent.



However, there is no precocongruence containing the pair $\langle t_1, s\rangle$. Suppose $R' \subseteq T \times S$ is an arbitrary precocongruence between $\mathcal{T}'$ and $\mathcal{S}$. Since $\langle \emptyset, \emptyset\rangle$ is R'-coherent, $\emptyset \in \nu'_T(t_2)$ and $\emptyset \notin \nu_S(s)$, it follows from Proposition 5.3.14 that $\langle t_2, s\rangle \notin R'$. This implies that $\langle \{t_2\}, \emptyset\rangle$ is R'-coherent, but $\{t_2\} \in \nu'_T(t_1)$ and $\emptyset \notin \nu_S(s)$, so $\langle t_1, s\rangle \notin R'$. ◁

To sum it up: Example 5.3.16 showed that precocongruences are a clear improvement when compared to $\mathcal{2}^{\mathcal{2}}$-bisimulations. Example 5.3.17, however,

demonstrates that precocongruences are still incomplete as a proof principle for behavioural equivalence over neighbourhood frames.

From Theorem 5.3.11 of the previous subsection, we know that on a single neighbourhood frame, precocongruences do capture behavioural equivalence. Using the results of this subsection it follows easily that, in fact, also $\mathcal{2}^{\mathcal{2}}$-bisimilarity captures behavioural equivalence on a single structure.

5.3.18. PROPOSITION. *If $\mathcal{S} = \langle S, \nu \rangle$ is a neighbourhood frame, and $R \subseteq S \times S$ is an equivalence relation, then:*

$$R \text{ is a } \mathcal{2}^{\mathcal{2}}\text{-bisimulation} \;\; \textit{iff} \;\; R \text{ is a precocongruence} \;\; \textit{iff} \;\; R \text{ is a congruence.}$$

Consequently, for all $s_1, s_2 \in S$: $\quad s_1 \sim s_2 \;\; \textit{iff} \;\; s_1 \sim_p s_2 \;\; \textit{iff} \;\; s_1 \sim_b s_2$.

PROOF. If $R \subseteq S \times S$ is an equivalence relation, then in particular $\mathrm{dom}(R) = \mathrm{rng}(R) = S$, and hence condition 1 of Proposition 5.3.13 is trivially satisfied. It follows from the characterisations in Propositions 5.3.13 and 5.3.14 that R is a $\mathcal{2}^{\mathcal{2}}$-bisimulation iff R is a precocongruence. The second equivalence is an instance of the more general result in Proposition 5.3.10(2). The final claim is an immediate consequence of the main claim and Lemma 2.3.3 (Chapter 2, p. 17). QED

5.3.19. REMARK. Alternatively, Proposition 5.3.18 follows from the result in [51] that congruences are F-bisimulations in case the functor F weakly preserves kernel pairs - a property that the functor $\mathcal{2}^{\mathcal{2}}$ has as the following argument shows: Let $f\colon S \to T$ be a function and consider its kernel $\ker(f) := \{\langle s, s' \rangle \in S \times S \mid f(s) = f(s')\}$ with projections $\pi_i\colon \ker(f) \to S$ for $i = 1, 2$. We have to show that for every pair of sets $N_1, N_2 \in \ker(\mathcal{2}^{\mathcal{2}}(f))$ there exists a set $N \in \mathcal{2}^{\mathcal{2}}(\ker(f))$ such that $\mathcal{2}^{\mathcal{2}}(\pi_i)(N) = N_i$ for $i = 1, 2$. Let N_1, N_2 be elements of $\mathcal{2}^{\mathcal{2}}(S)$ such that $\mathcal{2}^{\mathcal{2}}(f)(N_1) = \mathcal{2}^{\mathcal{2}}(f)(N_2)$. We put $N := \{\pi_1^{-1}(U_1) \mid U_1 \in N_1\} \cup \{\pi_2^{-1}(U_2) \mid U_2 \in N_2\}$. It is now easy to check that $\mathcal{2}^{\mathcal{2}}(\pi_i)(N) = N_i$ for $i = 1, 2$ as required. $\lhd$

5.3.3 Monotonic and Kripke bisimulations

Kripke frames and monotonic neighbourhood frames form subclasses of neighbourhood frames, as we have seen in Remarks 5.2.9 and 5.2.8, respectively. We now compare the different types of equivalence notions for neighbourhood frames to the existing bisimulation notions over these subclasses. Some of these results are well-known, but we repeat them here to give the reader an easy overview of how the different notions relate to each other.

The instantiation of cocongruences and precocongruences over monotonic neighbourhood frames and Kripke frames is made easy by the following lemma.

Given a natural transformation $\eta\colon F \dot{\to} G$ between Set-functors F and G, it is well-known (cf. [129]) that we obtain a functor $Tr_\eta\colon \mathsf{Coalg}(F) \to \mathsf{Coalg}(G)$ by composing with η: if $\langle X, \xi\rangle \in \mathsf{Coalg}(F)$ then $\langle X, \eta_{F(X)} \circ \nu\rangle \in \mathsf{Coalg}(G)$. This transformation preserves both bisimulations, cocongruences and precocongruences by naturality of η. We now show that if η is injective in all components, then it also reflects cocongruences and precocongruences. This result is an easy consequence of Lemma 3.8 in [15].

5.3.20. Lemma. *If $\eta\colon F \hookrightarrow G$ is a natural transformation that is injective in all components, i.e., F is a subfunctor of G, then for all F-coalgebras $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$, and all relations $R \subseteq X_1 \times X_2$:*

1. *R is an F-cocongruence between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ iff R is a G-cocongruence between $Tr_\eta(\langle X_1, \xi_1\rangle)$ and $Tr_\eta(\langle X_2, \xi_2\rangle)$.*

2. *R is an F-precocongruence between $\langle X_1, \xi_1\rangle$ and $\langle X_2, \xi_2\rangle$ iff R is a G-precocongruence between $Tr_\eta(\langle X_1, \xi_1\rangle)$ and $Tr_\eta(\langle X_2, \xi_2\rangle)$.*

Proof. Item 1 follows from [15, Lemma 3.8] and the fact that $\mathsf{Coalg}(G)$ has image factorisation for any Set-functor G. Item 2 follows directly from [15, Lemma 3.8], since the pushout arrows are always jointly surjective, i.e., $[p_1, p_2] : X_1 + X_2 \twoheadrightarrow P$, where $\langle P, p_1, p_2\rangle$ is the pushout of R. QED

Monotonic bisimulations

As mentioned in Remark 5.2.8, monotonic neighbourhood frames are coalgebras for the Set-functor $\mathcal{M}on$. The inclusion maps $\iota_X\colon \mathcal{M}on(X) \to \mathcal{2}^{\mathcal{2}}(X)$ yield a natural transformation $\iota\colon \mathcal{M}on \hookrightarrow \mathcal{2}^{\mathcal{2}}$ that is clearly injective in all components. It follows from Lemma 5.3.20 that a relation R is a $\mathcal{M}on$-cocongruence iff R is a $\mathcal{2}^{\mathcal{2}}$-cocongruence, and R is a $\mathcal{M}on$-precocongruence iff R is a $\mathcal{2}^{\mathcal{2}}$-precocongruence. Combining this last observation with Proposition 5.3.10(2), it follows that an equivalence relation R is a $\mathcal{M}on$-congruence iff R is a $\mathcal{2}^{\mathcal{2}}$-congruence.

Monotonic bisimulations were introduced by Pauly in [113].

5.3.21. Definition. (cf. [113]) Let $\mathcal{M}_1 = \langle S_1, \nu_1\rangle$ and $\mathcal{M}_2 = \langle S_2, \nu_2\rangle$ be monotonic neighbourhood frames, and $Z \subseteq S_1 \times S_2$ a relation. Z is a *monotonic bisimulation* between $\mathcal{M}_1$ and $\mathcal{M}_2$ if for all $\langle s_1, s_2\rangle \in Z$:

(mon-fwd) for all $U_1 \subseteq S_1$: if $U_1 \in \nu_1(s_1)$ then $Z[U_1] \in \nu_2(s_2)$.
(mon-bwd) for all $U_2 \subseteq S_2$: if $U_2 \in \nu_2(s_2)$ then $Z^{-1}[U_2] \in \nu_1(s_1)$.

Two states are monotonic bisimilar, if they are linked by a monotonic bisimulation. ◁

The following lemma relates *Mon*-precocongruences to these notions. We need to define the following property of relations. A relation $R \subseteq S_1 \times S_2$ is called *z-closed* (cf. [81]), if the following holds for all $s_1, t_1 \in S_1$ and $s_2, t_2 \in S_2$:

$$(z1) \quad \langle s_1, s_2\rangle, \langle s_1, t_2\rangle, \langle t_1, t_2\rangle \in R \;\Rightarrow\; \langle t_1, s_2\rangle \in R, \text{ and}$$
$$(z2) \quad \langle s_1, s_2\rangle, \langle t_1, s_2\rangle, \langle t_1, t_2\rangle \in R \;\Rightarrow\; \langle s_1, t_2\rangle \in R,$$

as illustrated here:

5.3.22. Lemma. *Let $\mathcal{S}_1 = \langle S_1, \nu_1\rangle$ and $\mathcal{S}_2 = \langle S_2, \nu_2\rangle$ be monotonic neighbourhood frames, and $R \subseteq S_1 \times S_2$ a relation.*

1. *If R is a Mon-bisimulation, then R is a monotonic bisimulation.*
2. *If R is a monotonic bisimulation, then R is a Mon-precocongruence.*
3. *If R is a z-closed Mon-precocongruence, then R is a monotonic bisimulation.*

Proof. Item 1 was shown in [56]. To see that item 2 holds, assume R is a monotonic bisimulation. Let $\langle s_1, s_2\rangle \in R$, $U_1 \subseteq S_1$, $U_2 \subseteq S_2$ be such that $\langle U_1, U_2\rangle$ is R-coherent, that is, $R[U_1] \subseteq U_2$ and $R^{-1}[U_2] \subseteq U_1$. It follows easily from monotonicity, (mon-fwd) and (mon-bwd) that $U_1 \in \nu_1(s_1)$ iff $U_2 \in \nu_2(s_2)$.

To prove item 3, assume R is a z-closed *Mon*-precocongruence, $\langle s_1, s_2\rangle \in R$, and let $U_1 \subseteq S_1$, $U_2 \subseteq S_2$ be arbitrary. Using the assumption that R is z-closed, it is easy to show that the pair $\langle U_1 \cup R^{-1}[R[U_1]], R[U_1]\rangle$ is R-coherent. Hence if $U_1 \in \nu_1(s_1)$ then by monotonicity, $U_1 \cup R^{-1}[R[U_1]] \in \nu_1(s_1)$ and since R is a *Mon*-precocongruence we get from the characterisation in Proposition 5.3.14 and R-coherence that $R[U_1] \in \nu_2(s_2)$. Similarly, it can be shown that $\langle R^{-1}[U_2], U_2 \cup R[R^{-1}[U_2]]\rangle$ is R-coherent, so if $U_2 \in \nu_2(s_2)$, then by monotonicity and R-coherence, it follows that $R^{-1}[U_2] \in \nu_1(s_1)$. QED

Mon does not preserve weak pullbacks (cf. [56]), so *Mon*-behavioural equivalence is strictly weaker than *Mon*-bisimilarity, but the other equivalence notions all coincide over monotonic neighbourhood frames.

5.3.23. Proposition. *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be monotonic neighbourhood frames containing states s_1 and s_2, respectively. The following are equivalent:*

1. *s_1 and s_2 are Mon-behaviourally equivalent,*
2. *s_1 and s_2 are monotonic bisimilar,*
3. *s_1 and s_2 are linked by a Mon-precocongruence.*

PROOF. (1 $\Leftrightarrow$ 2) was shown in [56], (2 $\Rightarrow$ 3) holds by Lemma 5.3.22(2), and (3 $\Rightarrow$ 1) follows from the general result in Proposition 5.3.8(2). QED

The next example shows that the z-closure requirement in Lemma 5.3.22(2) is really necessary.

5.3.24. EXAMPLE. Consider the two monotonic neighbourhood frames $\mathcal{M}_1 = \langle \{s_1, t_1, u_1\}, \nu_1 \rangle$ and $\mathcal{M}_2 = \langle \{s_2, t_2, u_2\}, \nu_2 \rangle$ where $\nu_1(s_1) = \uparrow\{t_1\}$, $\nu_2(s_2) = \uparrow\{t_2, u_2\}$, and $\nu_1(t_1) = \nu_1(u_1) = \nu_2(t_2) = \nu_2(u_2) = \uparrow\emptyset$. The relation $Z = \{\langle s_1, s_2\rangle, \langle t_1, t_2\rangle, \langle u_1, u_2\rangle, \langle u_1, t_2\rangle\}$ is a *Mon*-precocongruence. To see this, we use the characterisation from Proposition 5.3.14. First observe that the only non-trivial Z-coherent pair is $\langle \{t_1, u_1\}, \{t_2, u_2\}\rangle$. Since $\langle s_1, s_2\rangle \in Z$, we confirm (using monotonicity of ν_1) that $\{t_1, u_1\} \in \nu_1(s_1)$ iff $\{t_2, u_2\} \in \nu_2(s_2)$. The condition for the other states and Z-coherent pairs can be just easily checked. Using the characterisation in Proposition 5.3.13 we find that Z is also a $\mathcal{2}^{\mathcal{2}}$-bisimulation. However, Z is not a monotonic bisimulation, since $Z[\{t_1\}] = \{t_2\} \notin \nu_2(s_2)$. $\lhd$

Kripke bisimulations

Kripke frames are $\mathcal{P}$-coalgebras, and in Subsection 2.4.3 we saw that $\mathcal{P}$-bisimulations are the same as Kripke bisimulations. The transformation $(-)^{\mathrm{aug}}$ from Kripke frames to augmented neighbourhood frames (cf. Remark 5.2.9), in fact, arises from the natural transformation $\upsilon\colon \mathcal{P} \dot{\to} \mathit{Mon}$ defined for all sets X and $U \subseteq X$ by $\upsilon_X(U) = \uparrow U = \{V \subseteq X \mid U \subseteq V\}$. This υ is injective in all components. Hence, by Lemma 5.3.20, a relation R on a $\mathcal{P}$-coalgebra is a $\mathcal{P}$-cocongruence iff R is a *Mon*-cocongruence iff R is a $\mathcal{2}^{\mathcal{2}}$-cocongruence. Similarly, the notions of $\mathcal{P}$-precocongruences, *Mon*-precocongruences and $\mathcal{2}^{\mathcal{2}}$-precocongruences all coincide over $\mathcal{P}$-coalgebras, and from Proposition 5.3.10(2), we also obtain that on a $\mathcal{P}$-coalgebra, a relation R is $\mathcal{P}$-congruence iff R is a *Mon*-congruence iff R is a $\mathcal{2}^{\mathcal{2}}$-congruence.

5.3.25. PROPOSITION. *Let $\mathcal{K}_1 = \langle S_1, R_1\rangle$ and $\mathcal{K}_2 = \langle S_2, R_2\rangle$ be Kripke frames, and $R \subseteq S_1 \times S_2$ a relation.*

1. *R is a Kripke bisimulation between $\mathcal{K}_1$ and $\mathcal{K}_2$ iff R is a monotonic bisimulation between $\mathcal{K}_1^{aug}$ and $\mathcal{K}_2^{aug}$,*
2. *if R is a $\mathcal{P}$-cocongruence, then R is a Kripke bisimulation.*
3. *if R is a Kripke bisimulation, then R is a $\mathcal{P}$-precocongruence.*
4. *if R is a z-closed $\mathcal{P}$-precocongruence, then R is a Kripke bisimulation.*

PROOF. Item 1 was shown by Pauly in [113]. Item 2 follows from the fact that $\mathcal{P}$ preserves weak pullbacks (cf. [129, Theorem 4.3]). Item 3 is an instance

of Proposition 5.3.8(1). In order to prove item 4, suppose R is a z-closed $\mathcal{P}$-precocongruence. By Lemma 5.3.20(2), R is a z-closed *Mon*-precocongruence, hence by Lemma 5.3.22(3), a monotonic bisimulation. It now follows from item 1 that R is a Kripke bisimulation. QED

We note that the two frames in Example 5.3.24 are augmented, and this example therefore shows that even if a functor F preserves weak pullbacks, an F-precocongruence must be z-closed in order to yield an F-bisimulation, in general. Since the relation Z in Example 5.3.24 is also a 2^{2}-bisimulation, it shows that a 2^{2}-bisimulation must also be z-closed in order to be a guaranteed to be a Kripke bisimulation.

Due to Proposition 5.3.8. we conclude that all three equivalence notions for the functor $\mathcal{P}$ coincide.

5.3.26. PROPOSITION. *Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be Kripke frames containing states s_1 and s_2, respectively. The following are equivalent:*

1. *s_1 and s_2 are $\mathcal{P}$-behaviourally equivalent,*
2. *s_1 and s_2 are $\mathcal{P}$-bisimilar (i.e. Kripke bisimilar),*
3. *s_1 and s_2 are linked by a $\mathcal{P}$-precocongruence.*

PROOF. $(1 \Rightarrow 2)$ is well-known, and it follows from Lemma 5.3.25(2). The implications $(2 \Rightarrow 3)$ and $(3 \Rightarrow 1)$ follow from Proposition 5.3.8. QED

Apart from the correspondence between augmented neighbourhood frames and Kripke frames, any neighbourhood frame can be translated into a Kripke frame for a 3-modal language (cf. [78]). This language $\mathcal{L}^{\sharp}(\mathsf{At})$ has three unary modalities $\Box, [\ni]$ and $[\not\ni]$, and a neighbourhood frame $\mathcal{M} = \langle S, \nu \rangle$ can be viewed as a Kripke $\mathcal{L}^{\sharp}$-frame $\mathcal{M}^{\sharp} = \langle W, N, R_{\ni}, R_{\not\ni} \rangle$ by taking:

$$\begin{array}{lcl} W & = & S \cup \nu[S], \\ N & = & \{\langle s, U \rangle \in S \times \nu[S] \mid U \in \nu(s)\}, \\ R_{\ni} & = & \{\langle U, s \rangle \in \nu[S] \times S \mid s \in U\}, \\ R_{\not\ni} & = & \{\langle U, s \rangle \in \nu[S] \times S \mid s \notin U\}. \end{array}$$

The usual definition of 3-Kripke bisimulation applies to $\mathcal{L}^{\sharp}$-frames, that is, a relation Z is a 3-Kripke bisimulation if for each of the three relations $N, R_{\ni}, R_{\not\ni}$ the usual back-and-forth Kripke bisimulation conditions hold. Example 5.3.16 shows that 2^{2}-bisimilarity does not imply 3-Kripke bisimilarity, since $\mathrm{Gr}(f_1)$ is a 2^{2}-bisimulation and hence t_1 and u_1 are 2^{2}-bisimilar, but the forward condition for N clearly fails for any relation containing $\langle t_1, u_1 \rangle$ on the corresponding Kripke $\mathcal{L}^{\sharp}$-frames. The other implication, however, does hold. One can show that

given neighbourhood frames $\mathcal{M}_1 = \langle S_1, \nu_1 \rangle$ and $\mathcal{M}_2 = \langle S_2, \nu_2 \rangle$, and a 3-Kripke bisimulation $R \subseteq (S_1 \cup \nu_1[S_1]) \times (S_2 \cup \nu_2[S_2])$ between $\mathcal{M}_1^\sharp$ and $\mathcal{M}_2^\sharp$, then the state-part of R, $R \cap (S_1 \times S_2)$, is a $\mathcal{2}^{\mathcal{2}}$-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$. The proof is fairly straightforward, but also quite long and technical, and we have therefore chosen to leave it out.

5.4 Hennessy-Milner classes

The Hennessy-Milner theorem for normal modal logic states that over the class of finite Kripke models, two states are Kripke bisimilar if and only if they satisfy the same modal formulas. It is well-known (see e.g. [25]), that this Hennessy-Milner theorem can be generalised to hold over any class of modally saturated Kripke models, in particular, over the class of image-finite Kripke models.

In this section, we define modal saturation and image-finiteness for neighbourhood models and show that each of these properties leads to a Hennessy-Milner style theorem. However, in contrast with the case for Kripke models, we will see that image-finite neighbourhood models are not necessarily modally saturated. In the last subsection we describe ultrafilter extensions of neighbourhood models, and show that they are modally saturated.

First, we make precise what we mean by a Hennessy-Milner class of neighbourhood models. Since we have three equivalence notions for neighbourhood models, we have, in principle, three types of Hennessy-Milner classes. However, Examples 5.3.16 and 5.3.17 of section 5.3 showed that even over the class of finite neighbourhood models, two states can be behaviourally equivalent, and hence modally equivalent, without being linked by a precocongruence or a bisimulation. This means that precocongruences and bisimulations do not fit well with the expressivity of the modal language. We therefore define Hennessy-Milner classes with respect to behavioural equivalence.

5.4.1. Definition. A class $\mathbf{K}$ of neighbourhood models is a *Hennessy-Milner class*, if for any $\mathcal{M}_1$ and $\mathcal{M}_2$ in $\mathbf{K}$ containing states s_1 and s_2, respectively, we have: $\mathcal{M}_1, s_1 \equiv \mathcal{M}_2, s_2$ iff $\mathcal{M}_1, s_1 \sim_b \mathcal{M}_2, s_2$. ◁

The following lemma provides an easy, but useful, criterion for proving that a class of models is a Hennessy-Milner class.

5.4.2. Lemma. *Let $\mathbf{K}$ be a class of neighbourhood models. If for any $\mathcal{M}_1, \mathcal{M}_2 \in \mathbf{K}$, the modal equivalence relation $\equiv$ is a congruence on $\mathcal{M}_1 + \mathcal{M}_2$, then $\mathbf{K}$ is a Hennessy-Milner class.*

Proof. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be neighbourhood models in $\mathbf{K}$, and let $\iota_i \colon \mathcal{M}_i \to \mathcal{M}_1 + \mathcal{M}_2$ denote the canonical inclusion morphisms. Assume that we have

states s_1 and s_2 such that $\mathcal{M}_1, s_1 \equiv \mathcal{M}_2, s_2$. Since truth is invariant under bounded morphisms, we have $\iota_1(s_1) \equiv \iota_2(s_2)$ in $\mathcal{M}_1 + \mathcal{M}_2$. By assumption, $\equiv$ is a congruence on $\mathcal{M}_1 + \mathcal{M}_2$, hence $\varepsilon\colon \mathcal{M}_1 + \mathcal{M}_2 \to (\mathcal{M}_1 + \mathcal{M}_2)/{\equiv}$ is a bounded morphism, and $\langle s_1, s_2\rangle \in \mathrm{pb}(\varepsilon \circ \iota_1, \varepsilon \circ \iota_2)$, hence $s_1 \sim_b s_2$.

$$\begin{array}{ccccc} \mathcal{M}_1 & \xrightarrow{\iota_1} & \mathcal{M}_1 + \mathcal{M}_2 & \xleftarrow{\iota_2} & \mathcal{M}_2 \\ & & \downarrow{\scriptstyle \varepsilon} & & \\ & & (\mathcal{M}_1 + \mathcal{M}_2)/{\equiv} & & \end{array}$$

QED

5.4.1 Modally saturated models

In Lemma 5.4.2 we saw that in order to prove a Hennessy-Milner theorem, we are interested in neighbourhood models on which modal equivalence is a congruence. Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model. By applying the characterisations of congruences on neighbourhood frames in Corollary 5.3.15 and adding the condition for the atomic propositions, we find that $\equiv$ is a congruence on $\mathcal{M}$ iff for all $s, t \in S$ such that $s \equiv t$:

$$\begin{array}{lll} \text{(c1)} & \text{for all } p \in \mathsf{At}: & s \in V(p) \iff t \in V(p), \text{ and} \\ \text{(c2)} & \text{for all modally coherent } X \subseteq S: & X \in \nu(s) \iff X \in \nu(t). \end{array} \tag{5.9}$$

Clearly, condition (c1) holds in all neighbourhood models, since modally equivalent states must make the same atomic propositions true. One way of making condition (c2) hold, is to ensure that all modally coherent neighbourhoods are definable.

5.4.3. Lemma. *Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model. If for all $s \in S$ and all modally coherent $X \in \nu(s)$, there exists a modal $\mathcal{L}$-formula φ such that $X = [\![\varphi]\!]^{\mathcal{M}}$, then modal equivalence is a congruence on $\mathcal{M}$.*

Proof. Let X be a modally coherent neighbourhood of some state, and assume $X = [\![\varphi]\!]^{\mathcal{M}}$. We have for any $s, t \in S$ such that $s \equiv t$: $X \in \nu(s)$ iff $\mathcal{M}, s \models \Box\varphi$ iff $\mathcal{M}, t \models \Box\varphi$ iff $X \in \nu(t)$. QED

For finite models, a standard argument shows that any modally coherent neighbourhood X is definable by a formula of the form $\delta = \bigvee_{i\leq n} \bigwedge_{j\leq k} \delta_{i,j}$ where $n, k < \omega$. For infinite models, the same argument would yield a formula with an infinite disjunction and conjunction, which is not a well-formed formula of our finitary language. Modal saturation is a compactness property which allows us to replace infinite conjunctions and disjunctions with finite ones. Thus we can essentially use the same argument as for finite models to show that modally coherent neighbourhoods are definable (and we do so in Lemma 5.4.5 below).

We will use the following notation. Let Ψ be a set of modal $\mathcal{L}$-formulas and $\mathcal{M} = \langle S, \nu, V\rangle$ a neighbourhood model. We define $\neg\Psi = \{\neg\psi \mid \psi \in \Psi\}$, $[\![\bigwedge \Psi]\!]^{\mathcal{M}} = \bigcap_{\psi\in\Psi}[\![\psi]\!]^{\mathcal{M}}$, and $[\![\bigvee \Psi]\!]^{\mathcal{M}} = \bigcup_{\psi\in\Psi}[\![\psi]\!]^{\mathcal{M}}$. A set Ψ of $\mathcal{L}$-formulas is *satisfiable in a subset* $X \subseteq S$ of $\mathcal{M}$, if $[\![\bigwedge \Psi]\!]^{\mathcal{M}} \cap X \neq \emptyset$. A set Ψ of $\mathcal{L}$-formulas is *finitely satisfiable in* $X \subseteq S$, if any finite subset $\Psi_0 \subseteq_\omega \Psi$ is satisfiable in X.

5.4.4. Definition. Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model. A subset $X \subseteq S$ is called *modally compact* if for all sets Ψ of modal $\mathcal{L}$-formulas, Ψ is satisfiable in X whenever Ψ is finitely satisfiable in X. The neighbourhood model $\mathcal{M}$ is *modally saturated*, if for all $s \in S$ and all modally coherent neighbourhoods $X \in \nu(s)$, both X and the complement X^c are modally compact. ◁

To see why modal compactness is really a compactness property, note that for a subset X in a neighbourhood model $\mathcal{M}$, $X \subseteq [\![\bigvee \Psi]\!]^{\mathcal{M}}$ iff $\{\neg\psi \mid \psi \in \Psi\}$ is not satisfiable in X. Hence X is modally compact, if and only if, for all $\Psi \subseteq \mathcal{L}$ such that $X \subseteq [\![\bigvee \Psi]\!]^{\mathcal{M}}$ there is a $\Psi_0 \subseteq_\omega \Psi$ such that $X \subseteq [\![\bigvee \Psi_0]\!]^{\mathcal{M}}$. Clearly, any finite set is modally compact. Note also that, in Definition 5.4.4, due to the fact that $[\![\bigwedge \Psi]\!]^{\mathcal{M}} \subseteq X$ if and only if $X^c \subseteq [\![\bigvee \neg\Psi]\!]^{\mathcal{M}}$, we have that X^c is modally compact, if and only if, for all $\Psi \subseteq \mathcal{L}$ such that $[\![\bigwedge \Psi]\!]^{\mathcal{M}} \subseteq X$, there is a $\Psi_0 \subseteq_\omega \Psi$ such that $[\![\bigwedge \Psi_0]\!]^{\mathcal{M}} \subseteq X$.

5.4.5. Lemma. *Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a modally saturated neighbourhood model. For all $X \subseteq S$: X is modally coherent iff X is definable by a modal $\mathcal{L}$-formula.*

Proof. If $X = [\![\varphi]\!]^{\mathcal{M}}$ for some $\varphi \in \mathcal{L}$, then clearly X is modally coherent. For the converse implication, assume X is modally coherent, i.e., X is a union of modal equivalence classes $X = \bigcup_{c\in C}[x_c]_{\equiv}$. For $c \in C$ and $y \not\equiv x_c$ there is a modal $\mathcal{L}$-formula $\delta_{c,y}$ such that $x_c \models \delta_{c,y}$ and $y \models \neg\delta_{c,y}$, so by taking $\Delta_c = \{\delta_{c,y} \mid y \not\equiv x_c\}$, we have $[x_c]_{\equiv} = [\![\bigwedge \Delta_c]\!]^{\mathcal{M}} \subseteq X$ for each $c \in C$. By modal compactness of X^c, for each $c \in C$ there is a finite subset $\Delta_c^0 \subseteq_\omega \Delta_c$ such that $[x_c]_{\equiv} \subseteq [\![\bigwedge \Delta_c^0]\!]^{\mathcal{M}} \subseteq X$. Defining $\delta_c = \bigwedge \Delta_c^0$ for each $c \in C$, we therefore have $X = \bigcup_{c\in C}[\![\delta_c]\!]^{\mathcal{M}}$. Now by modal compactness of X, we get a finite subset $\Delta_0 \subseteq_\omega \{\delta_c \mid c \in C\}$ such that $X = [\![\bigvee \Delta_0]\!]^{\mathcal{M}}$. That is, X is definable by the formula $\delta = \bigvee \Delta_0$. QED

5.4.6. Proposition. *If $\mathcal{M}$ is a modally saturated neighbourhood model, then modal equivalence is a congruence on $\mathcal{M}$. It follows that modally equivalent states in $\mathcal{M}$ are behaviourally equivalent.*

Proof. Immediate consequence of Lemmas 5.4.3 and 5.4.5. QED

5.4.7. Corollary. *The class of finite neighbourhood models is a Hennessy-Milner class.*

PROOF. Since the disjoint union of two finite neighbourhood models is again finite, it suffices by Lemma 5.4.2 and Proposition 5.4.6 to show that finite neighbourhood models are modally saturated. But this is immediate, since any set of states in a finite neighbourhood model $\mathcal{M}$, is necessarily finite, and hence modally compact, so $\mathcal{M}$ is modally saturated. QED

The question remains whether the class of all modally saturated neighbourhood models is a Hennessy-Milner class. We conjecture that if $\mathcal{M}$ and $\mathcal{N}$ are modally saturated then modal equivalence is a congruence on $\mathcal{M} + \mathcal{N}$. If this is the case, then the Hennessy-Milner theorem follows from Lemma 5.4.2.

5.4.8. REMARK. In [113] the following definition of modal saturation for monotonic neighbourhood models was introduced, and it was shown that over the class of modally saturated monotonic neighbourhood models, modal equivalence implies monotonic bisimilarity. A monotonic neighbourhood model $\langle S, \nu, V\rangle$ is *monotonic modally saturated*, if for all $s \in S$ and all sets Ψ of modal $\mathcal{L}$-formulas the following hold:

(m1-mon) For all $X \in \nu(s)$, if Ψ is finitely satisfiable in X, then Ψ is satisfiable in X.

(m2-mon) If for all $\Psi_0 \subseteq_\omega \Psi$, there is an $X \in \nu(s)$ such that $X \subseteq (\bigwedge \Psi_0)$, then there is an $X \in \nu(s)$ such that $X \subseteq (\bigwedge \Psi)$.

In a monotonic neighbourhood model $\mathcal{M}$, (m1-mon) clearly implies that all modally coherent neighbourhoods are modally compact. The converse also holds, since for any neighbourhood X of some state s, the closure X' of X with respect to modal equivalence, i.e., $X' = \bigcup_{x\in X}[x]_\equiv$, is also a neighbourhood of s by monotonicity, and for any $\Psi \subseteq \mathcal{L}$, Ψ is satisfiable in X if and only if Ψ is satisfiable in X'. However, it is not clear whether monotonic modal saturation and (neighbourhood) modal saturation coincide in all monotonic models. We suspect that neither implies the other due to the following. The condition (m2-mon) says that all neighbourhood collections are closed under arbitrary intersections of definable neighbourhoods, a property which we expect can be shown to fail in some modally saturated neighbourhood model. On the other hand, it is not clear why the complements of modally coherent neighbourhoods should be modally compact in a monotonic modally saturated model. Unfortunately, at the moment we have no examples that confirm these intuitions. ◁

5.4.9. REMARK. A Kripke model $\langle S, R, V\rangle$ is *Kripke modally saturated*, if for all $s \in S$ and all sets Ψ of modal $\mathcal{L}$-formulas:

(m1-krip) If Ψ is finitely satisfiable in $R[s]$, then Ψ is satisfiable in $R[s]$,

and over the class of modally saturated Kripke models, modal equivalence implies Kripke bisimilarity (see e.g. [25]). From the above definitions, it is clear that for any augmented neighbourhood model $\mathcal{M}$, if $\mathcal{M}$ is monotonic modally saturated or (neighbourhood) modally saturated, then $\mathcal{M}^{\mathrm{krp}}$ is Kripke modally saturated. However, if $\mathcal{M}^{\mathrm{krp}}$ is Kripke modally saturated, then modally coherent neighbourhoods may fail to be modally compact in $\mathcal{M}$. This is shown by Example 5.4.17 (page 155) in the next subsection. Hence Kripke modal saturation does not imply monotonic modal saturation nor (neighbourhood) modal saturation. Note that (m2-mon) holds over any augmented neighbourhood model. ◁

As we have seen in Remarks 5.4.8 and 5.4.9, the notions of neighbourhood, monotonic and Kripke modal saturation do not restrict in a natural way. Moreover, in the next subsection (Example 5.4.17), we will see that image-finite neighbourhood models are not necessarily modally saturated. These observations could be interpreted as arguments for saying that our definition of modal saturation for neighbourhood models is not the right one. On the other hand, Definition 5.4.4 arises in a natural manner, it implies Kripke modal saturation over Kripke models, in subsection 5.4.3 we show that ultrafilter extensions of neighbourhood models are modally saturated, and in subsection 5.5.2 we will see that when viewing neighbourhood models as first-order models, then ω-saturation implies modal saturation (Lemma 5.5.6). We believe these are good arguments for Definition 5.4.4 being the right notion after all. However, further investigations are needed to support this claim. It would be useful to have a better understanding of what an abstract notion of modal saturation for F-coalgebras should be.

5.4.2 Image-finite neighbourhood models

In this section, we define image-finite neighbourhood models and prove that they form a Hennessy-Milner class. This result is, in fact, an instance of a more general result in coalgebraic modal logic (by Schröder [136]), as we briefly explain in Remark 5.4.15. Since neighbourhood structures are of general interest outside the world of coalgebra, we have chosen to present a self-contained proof, which can be understood and verified without the knowledge of coalgebraic modal logic. However, we hope that this section also illustrates the benefits of universal coalgebra, and that it may inspire readers to study the more general results in [136].

In contrast with the Kripke case, image-finite neighbourhood models are not necessarily modally saturated as we will see later in Example 5.4.17. Instead, we will show that they satisfy the condition of the following lemma.

5.4.10. Lemma. *Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model. If for any states $s_1, s_2 \in S$ and any modally coherent subset $X \subseteq S$ there is a formula $\varphi \in \mathcal{L}$ such that for any $i \in \{1, 2\}$, $X \in \nu(s_1)$ if and only if $[\![\varphi]\!]^{\mathcal{M}} \in \nu(s_2)$, then modal equivalence is a congruence on $\mathcal{M}$.*

Proof. Immediate by the characterisation given by conditions (c1) and (c2) on page 147. QED

A Kripke model is image-finite if every state has only finitely many successors. For neighbourhood models, the notion of image-finiteness is less obvious, but as with bisimilarity, universal coalgebra provides us with an abstract notion of image-finiteness for coalgebras which we instantiate for the $\mathcal{2}^{\mathcal{2}}$-functor. The general construction behind this definition is that of taking the finitary part of a functor. Recall that we denote the inclusion map of $Y \subseteq X$ by $\iota_Y \colon Y \hookrightarrow X$. Given any functor $F \colon \mathsf{Set} \to \mathsf{Set}$, define the functor F_ω by letting

$$F_\omega(X) = \bigcup\{F(\iota_Y)[FY] \mid \iota_Y \colon Y \hookrightarrow X, Y \subseteq_\omega X\}$$

for a set X, and for a function $f \colon X \to Y$, $F_\omega(f)$ is the restriction of $F(f)$ to $F_\omega(X)$. It is known that F_ω is the unique finitary (or ω-accessible) subfunctor of F which agrees with F on all finite sets (see e.g. [4, 110]), and F_ω is called the finitary part of F. We now give a characterisation of the finitary part of $\mathcal{2}^{\mathcal{2}}$. For a subset inclusion map $\iota_B : B \hookrightarrow X$ and $D \subseteq X$, note that $\iota_B^{-1}[D] = D \cap B$. If $U \in \mathcal{2}^{\mathcal{2}}{}_\omega(X)$ and $B \subseteq X$ is such that for all $D \subseteq X$: $D \in U \iff D \cap B \in U$, then we call B a *base set for* U.

5.4.11. Lemma. *Let X be a set. We have:*

$$\mathcal{2}^{\mathcal{2}}{}_\omega(X) = \{U \in \mathcal{2}^{\mathcal{2}}(X) \mid \exists B \subseteq_\omega X. \forall D \subseteq X : (D \in U \iff D \cap B \in U)\}.$$

Proof. Let X be a set and $U \in \mathcal{2}^{\mathcal{2}}(X)$. We have:

$$\begin{aligned} U \in \mathcal{2}^{\mathcal{2}}{}_\omega(X) &\iff \exists B \subseteq_\omega X : U \in \mathcal{2}^{\mathcal{2}}(\iota_B)[\mathcal{2}^{\mathcal{2}}(B)] \\ &\iff \exists B \subseteq_\omega X\, \exists V \in \mathcal{2}^{\mathcal{2}}(B) : U = \mathcal{2}^{\mathcal{2}}(\iota_B)(V) \\ &\iff \exists B \subseteq_\omega X\, \exists V \in \mathcal{2}^{\mathcal{2}}(B) : U = \{D \subseteq X \mid D \cap B \in V\} \\ &\iff \exists B \subseteq_\omega X\, \forall D \subseteq X : (D \in U \iff D \cap B \in U). \end{aligned}$$

QED

5.4.12. Definition. We define the class of *image-finite neighbourhood frames* as the class $\mathsf{Coalg}(\mathcal{2}^{\mathcal{2}}{}_\omega)$ of $\mathcal{2}^{\mathcal{2}}{}_\omega$-coalgebras. The class of *image-finite neighbourhood models* is the class of neighbourhood models based on an image-finite neighbourhood frame. ◁

So, image-finite neighbourhood frames are the neighbourhood frames in which all neighbourhood collections are determined by a finite base set. It should be clear that a finite neighbourhood frame $\langle S, \nu \rangle$ is image-finite, since for all $s \in S$, S is a finite base set for $\nu(s)$. In proving that image-finite neighbourhood models form a Hennessy-Milner class, we use the following lemma.

5.4.13. Lemma. *Let S be a set and θ an equivalence relation on S. Moreover, let $B \subseteq S$ and denote by $B_\theta \subseteq B$ a set of representatives of the θ-classes intersecting B. For all $X, X' \subseteq S$, if X and X' are both θ-coherent, then $X \cap B = X' \cap B$ iff $X \cap B_\theta = X' \cap B_\theta$.*

Proof. Let S, B and $B_\theta \subseteq B$ be as stated, and assume that X and X' are θ-coherent subsets of S. It is clear that $X \cap B = X' \cap B$ implies $X \cap B_\theta = X' \cap B_\theta$. For the other implication, assume $X \cap B_\theta = X' \cap B_\theta$. We have: $s \in X \cap B$ implies there is an $s' \in B_\theta$ such that $s \theta s'$. Since X is θ-coherent, $s' \in X \cap B_\theta = X' \cap B_\theta$. Now since X' is θ-coherent, $s \in X'$, and thus $s \in X' \cap B$. Hence we have shown $X \cap B \subseteq X' \cap B$. The other inclusion is shown similarly. QED

5.4.14. Proposition. *The class of image-finite neighbourhood models is a Hennessy-Milner class.*

Proof. For any functor F, the category $\mathsf{Coalg}(F)$ has coproducts (cf. Definition 2.2.6, p. 14). It follows that image-finite neighbourhood models are closed under disjoint unions. By Lemma 5.4.2 it suffices to show that in an image-finite neighbourhood model, modal equivalence is a congruence. So let $\mathcal{M} = \langle S, \nu, V \rangle$ be image-finite, and let $s, t \in S$. We then have finite base sets $B_s, B_t \subseteq_\omega S$ for $\nu(s)$ and $\nu(t)$, respectively. Let $B_{st} = B_s \cup B_t$. By Lemma 5.4.10 it suffices to find for any modally coherent $X \subseteq S$, a formula $\varphi \in \mathcal{L}$ such that

$$X \cap B_{st} \;=\; [\![\varphi]\!]^{\mathcal{M}} \cap B_{st}, \tag{5.10}$$

since then $X \cap B_s = [\![\varphi]\!]^{\mathcal{M}} \cap B_s$ and $X \cap B_t = [\![\varphi]\!]^{\mathcal{M}} \cap B_t$, and hence $X \in \nu(s)$ iff $[\![\varphi]\!]^{\mathcal{M}} \in \nu(s)$, similarly for t, and consequently, if $s \equiv t$, then $X \in \nu(s)$ if and only if $X \in \nu(t)$.

We now show how to obtain such a φ. Let $X \subseteq$ be modally coherent and let $B'_{st} \subseteq B_{st}$ be a set of representatives of the $\equiv$-classes intersecting B_{st}. Since B_{st} is finite, so is B'_{st}. Assume $B'_{st} = \{s_1, \ldots, s_n\}$. Now there are modal formulas $\varphi_1, \ldots, \varphi_n \in \mathcal{L}$ which characterise $s_1, \ldots, s_n$, respectively, within B'_{st}, that is, $\mathcal{M}, s_i \models \varphi_j$ iff $i = j$, for $1 \leq i, j \leq n$. Namely, for each $s_i \in B'_{st}$, we have for all $s_j \in B'_{st} \setminus \{s_i\}$, $s_i \not\equiv s_j$. Hence there is a formula $\varphi_{i,j}$ such that $\mathcal{M}, s_i \models \varphi_{i,j}$ and $\mathcal{M}, s_j \not\models \varphi_{i,j}$. Take $\varphi_i = \bigwedge_{j=1, j \neq i}^{n} \varphi_{i,j}$, $i = 1, \ldots, n$. We now define $\varphi = \bigvee \{\varphi_i \mid s_i \in X \cap B'_{st}\}$. To see that φ satisfies (5.10) it suffices by Lemma 5.4.13 to show that $X \cap B'_{st} = [\![\varphi]\!]^{\mathcal{M}} \cap B'_{st}$. Clearly, by definition of

φ, if $s_i \in X \cap B'_{st}$ then $s_i \in [\![\varphi]\!]^{\mathcal{M}} \cap B'_{st}$. Conversely, if $s_j \in [\![\varphi]\!]^{\mathcal{M}} \cap B'_{st}$ then $\mathcal{M}, s_j \models \varphi_i$ for some i such that $s_i \in X \cap B'_{st}$. Since φ_i characterises s_i in B'_{st}, it follows that $s_j = s_i \in X \cap B'_{st}$. QED

5.4.15. Remark. As we already mentioned, Proposition 5.4.14 is a consequence of a more general result in coalgebraic modal logic, which we briefly explain here. In coalgebraic modal logic, the semantics of modalities is given by predicate liftings. A predicate lifting for a functor $F\colon \mathsf{Set} \to \mathsf{Set}$ is a natural transformation $\lambda\colon 2 \to 2 \circ F$. Given a set Λ of predicate liftings for F, the finitary coalgebraic modal language $\mathcal{L}(\Lambda)$ is the multi-modal language which contains a modality $[\lambda]$ for each $\lambda \in \Lambda$. Given an F-coalgebra $\mathbb{X} = \langle X, \xi \rangle$, the truth of formulas is defined in the standard inductive manner for the basic Boolean connectives. The truth of a modal formula $[\lambda]\phi$ is defined by: $\mathbb{X}, x \models [\lambda]\phi$ iff $\xi(x) \in \lambda_X([\![\phi]\!]^{\mathbb{X}})$. Atomic propositions can also be interpreted using constant predicate liftings. We refer to [111] for details.

Using currying, every predicate lifting $\lambda\colon 2 \to 2 \circ F$ corresponds to a natural transformation $\hat{\lambda}\colon F \to 2^2$, called the transposite of λ. A set Λ of predicate liftings for F is called *separating* if the source of transposites $\{\hat{\lambda} \mid \lambda \in \Lambda\}$ is jointly injective. Schröder shows in [136, Theorem 41, Corollary 45]) that if $F\colon \mathsf{Set} \to \mathsf{Set}$ is finitary and Λ is separating, then the finitary coalgebraic modal language $\mathcal{L}(\Lambda)$ is expressive for F-coalgebras, meaning that over the class of F-coalgebras, $\mathcal{L}(\Lambda)$-equivalence implies behavioural equivalence.

We can instantiate this result for the finitary functor ${2^2}_\omega \times \mathcal{P}(\mathsf{At})$ and classical modal logic. The basic modal language and its interpretation over neighbourhood models is the finitary coalgebraic modal logic given by $\Lambda = \{\lambda\} \cup \{\rho_i \mid i < \omega\}$, where $\lambda\colon 2 \to 2 \circ {2^2}_\omega$ is defined by $\lambda_X(A) = \{U \in {2^2}_\omega(X) \mid A \in U\}$, and the ρ_i, $i < \omega$, are constant predicate liftings that interpret the atomic propositions. It is known that $\{\lambda\} \cup \{\rho_i \mid i < \omega\}$ is separating iff $\{\lambda\}$ is separating. The transposite $\hat{\lambda}\colon {2^2}_\omega \to 2^2$ is simply the inclusion map, i.e., $\hat{\lambda}_X = \iota_{{2^2}_\omega(X)}$ for all sets X, so trivially $\{\hat{\lambda}\}$ is jointly injective, hence $\{\lambda\}$ is separating. It now follows from Schröder's results that over the class of image-finite neighbourhood models, modal equivalence implies behavioural equivalence. ◁

We now show that the notion of image-finiteness for neighbourhood frames restricts to the subclasses of neighbourhood frames that correspond with Kripke frames and monotonic neighbourhood frames, respectively.

Monotonic neighbourhood frames are coalgebras for the subfunctor Mon of 2^2 (cf. Remark 5.2.8) which sends a set X to the collection of all subsets of $\mathcal{P}(X)$ which are closed under supersets. Due to motonicity, given a function $f\colon X \to Y$, we can describe $\mathit{Mon}(f)$ in terms of the direct image of f, namely, for all $V \in \mathit{Mon}(X)$, $\mathit{Mon}(f)(V) = \bigcup\{\uparrow f[D] \mid D \in V\}$. Recall that for a subset $B \subseteq X$, $\uparrow B = \{B' \subseteq X \mid B \subseteq B'\}$. Image-finite monotonic neighbourhood frames,

are then nothing but Mon_ω-coalgebras. Let X be a set and $U \in Mon(X)$. We have:

$$\begin{array}{rcl} U \in Mon_\omega(X) & \iff & \exists Y \subseteq_\omega X \, \exists V \in Mon(Y) : U = Mon(\iota_Y)(V) \\ & \iff & \exists Y \subseteq_\omega X \, \exists V \in Mon(Y) : U = \bigcup\{\uparrow \iota_Y[B] \mid B \in V\} \\ & \iff & \exists C_1, \ldots, C_n \subseteq_\omega X : U = \uparrow C_1 \cup \ldots \cup \uparrow C_n. \end{array}$$

The neighbourhood collections in an image-finite monotonic neighbourhood model are thus generated by finite sets of finite neighbourhoods which are minimal with respect to $\subseteq$ in $\mathcal{P}(X)$. Such minimal neighbourhoods will be referred to as core neighbourhoods. More precisely, if $\mathcal{M} = \langle S, \nu, V\rangle$ is a neighbourhood model, $s \in S$ and $C \in \nu(s)$ is such that for all $D \subsetneq C$, $D \notin \nu(s)$, C is called a *core neighbourhood* of s. The collection of core neighbourhoods of s is denoted $\nu^c(s)$. This terminology follows [114, 52] where image-finite monotonic neighbourhood models were called *locally core finite*.

Finally, recall that a Kripke model $\langle S, R, V\rangle$ is image-finite (cf. [25]), if for all $s \in S$, the set of R-successors $R[s]$ is finite.

5.4.16. Proposition. *Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model.*

1. *If $\mathcal{M}$ is a monotonic neighbourhood model, then $\mathcal{M}$ is image-finite as a monotonic neighbourhood model iff $\mathcal{M}$ is image-finite as a neighbourhood model.*

2. *If $\mathcal{M}$ is augmented, then $\mathcal{M}^{krp}$ is image-finite as a Kripke model iff $\mathcal{M}$ is image-finite as a neighbourhood model.*

Proof. To prove item 1, let $\mathcal{M}$ be monotonic. Since Mon is a subfunctor of $\mathcal{2}^{\mathcal{2}}$, also Mon_ω is a subfunctor of $\mathcal{2}^{\mathcal{2}}_\omega$. It follows that any image-finite monotonic model is also image-finite as a neighbourhood model. Concretely, one can show that for all $s \in S$, the union of core neighbourhoods $B = \bigcup \nu^c(s)$ is a finite base set for $\nu(s)$. For the other direction, assume $\mathcal{M}$ is image-finite as a neighbourhood model. Let $s \in S$, and assume $B \subseteq_\omega S$ is a finite base set for $\nu(s)$. We first show that every neighbourhood is in the upwards closure of some finite core neighbourhood: $U \in \nu(s)$ implies $B \cap U \in \nu(s)$, and since $B \cap U$ is finite, there must be a finite $C \in \nu^c(s)$ such that $C \subseteq B \cap U \subseteq U$. Suppose now that $C \in \nu^c(s)$ is an arbitrary core neighbourhood of s. As B is a base set for $\nu(s)$, $C \cap B \in \nu(s)$, and hence by $\subseteq$-minimality of C, $C \subseteq B$. It now follows from the finiteness of B, that s has only finitely many core neighbourhoods $C_1, \ldots, C_n$ of finite cardinality, and $\nu(s) = \uparrow C_1 \cup \ldots \cup \uparrow C_n$.

For item 2, let $\mathcal{M}^{\mathrm{krp}} = \langle S, R, V\rangle$, i.e., for all $s \in S$, $\nu(s) = \uparrow R[s]$, and $\nu^c(s) = \{R[s]\}$. This immediately shows that if $\mathcal{M}^{\mathrm{krp}}$ is image-finite then $\mathcal{M}$ is image-finite as a monotonic model, and hence by item 1, also as a neighbourhood model. Conversely, if $\mathcal{M}$ is image-finite, then by item 1, $\mathcal{M}$ is image-finite as a monotonic model, hence for all $s \in S$, $\bigcup \nu^c(s) = R[s]$ is finite. QED

The following example demonstrates that image-finite neighbourhood models are not necessarily modally saturated, and it also shows that a Kripke modally saturated model, is not necessarily modally saturated as a (monotonic) neighbourhood model.

5.4.17. Example. Consider the Kripke model $\mathcal{K} = \langle S, R, V\rangle$ where $S = \mathbb{N}$, the set of natural numbers, and R is the usual relation $>$ on $\mathbb{N}$, that is, for $m, n \in \mathbb{N}$, $\langle m, n\rangle \in R$ iff $m > n$, and $R[m] = \{n \in \mathbb{N} \mid n < m\}$. Finally, the valuation V is defined as $V(p_i) = \emptyset$, for all atomic propositions $p_i \in \mathsf{At}$. $\mathcal{K}$ is an image-finite Kripke model, hence by Proposition 5.4.16 the augmented neighbourhood model $\mathcal{K}^{\mathrm{aug}}$ corresponding to $\mathcal{K}$ is also image-finite as a (monotonic) neighbourhood model. Since $\mathcal{K}$ is image-finite, $\mathcal{K}$ is Kripke modally saturated. However, $\mathcal{K}^{\mathrm{aug}}$ is not modally saturated as a neighbourhood model nor as a monotonic model. To see this, first note that the set $\mathbb{N}$ is trivially modally coherent and by monotonicity $\mathbb{N}$ is also a neighbourhood of every $n \in \mathbb{N}$. Now, consider the set of modal $\mathcal{L}$-formulas, $\Psi = \{\Diamond^n\Box\bot \mid n \in \mathbb{N}\}$. Note that by transitivity, $\mathcal{K}, m \models \Diamond^n\Box\bot$ iff $m \geq n$. Since $\mathcal{K}$ and $\mathcal{K}^{\mathrm{aug}}$ are pointwise equivalent, and every finite subset $\Psi_0 \subseteq_\omega \Psi$ is satisfiable in $\mathcal{K}$ at the maximal $n \in \mathbb{N}$ such that $\Diamond^n\Box\bot \in \Psi_0$, it follows that Ψ is finitely satisfiable in the neighbourhood $\mathbb{N}$ in $\mathcal{K}^{\mathrm{aug}}$. However, Ψ is clearly not satisfiable in $\mathbb{N}$. We have thus shown that $\mathbb{N}$ is not modally compact, hence $\mathcal{K}^{\mathrm{aug}}$ is not (monotonic) modally saturated. $\lhd$

5.4.3 Ultrafilter extensions

In this section, we prove a behavioural-equivalence-somewhere-else result by showing that any two modally equivalent states of neighbourhood models have behaviourally equivalent representatives in the ultrafilter extensions of these neighbourhood models. To this end, we define ultrafiler extensions of neighbourhood models, and we prove analogues of results known for ultrafilter extensions of Kripke models. In particular, we show that ultrafilter extensions are modally saturated. This result will be used in our proof of Craig interpolation in subsection 5.5.3.

Just as ultrafilter extensions of Kripke models are obtained from algebraic duality (see e.g. [25]), ultrafilter extensions of neighbourhood models are a by-product of a more general duality between coalgebras and certain algebras on the category of Boolean algebras, as described in e.g. [83, 89]. Our definition of ultrafilter extensions of neighbourhood frames is obtained by instantiating the more general definition of ultrafilter extensions of F-coalgebras presented in [89] to $F = \mathcal{2}^{\mathcal{2}}$. The basic properties follow from the category theoretical framework. With quite some effort, the behavioural-equivalence-somewhere-else result can be obtained as a special case of a more general theorem in [83]. However, instead of requiring knowledge of the (rather abstract) theory in [83, 89], we

have chosen to give a direct, concrete description of ultrafilter extensions of neighbourhood models, and to use standard model-theoretic techniques to prove basic properties. We believe that such a presentation will make the results of this section and the proof of the Craig interpolation theorem better accessible to readers whose background is mainly in modal logic. For the interested reader, we give a brief summary of the construction from [89] in Remark 5.4.20.

Let us begin by introducing some terminology and notation, and recalling some facts concerning ultrafilters.

5.4.18. Definition. Let $\mathcal{A} = \langle A, \wedge, \vee, -, 0, 1\rangle$ be a Boolean algebra with the usual ordering: for all $a, b \in A$: $a \leq b$ iff $a \wedge b = a$. A subset $\mathbf{u} \subseteq A$ is an *ultrafilter of* $\mathcal{A}$ if $1 \in \mathbf{u}$, and for all $a, b \in A$: $a, b \in \mathbf{u}$ implies $a \wedge b \in \mathbf{u}$, $a \in \mathbf{u}$ and $a \leq b$ implies $b \in \mathbf{u}$, and $a \in \mathbf{u}$ iff $-a \notin \mathbf{u}$.

Let S be a set. The collection of *ultrafilters over* S, denoted by $\mathrm{Uf}(S)$, is the set of ultrafilters of the Boolean powerset algebra $\langle \mathcal{P}(S), \cap, \cup, \setminus, \emptyset, S\rangle$. For $U \subseteq S$ and $s \in S$, we define

$$\begin{aligned} \hat{U} &:= \{\mathbf{u} \in \mathrm{Uf}(S) \mid U \in \mathbf{u}\}, \\ \mathsf{u}_s &:= \{V \subseteq S \mid s \in V\}. \end{aligned}$$

It is easily confirmed that u_s is an ultrafilter over S; u_s is called the *principal ultrafilter* generated by s. The induced map $\mathsf{u}\colon S \to \mathrm{Uf}(S)$ is called the *principal ultrafilter map*. ◁

The duality betwen Stone spaces and Boolean algebras gives rise to the following two contravariant functors. $\mathbb{P}\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{BA}$ maps a set X to its Boolean algebra of subsets. The functor $\mathbb{U}\colon \mathsf{BA} \to \mathsf{Set}^{\mathrm{op}}$ maps a Boolean algebra to the set of its ultrafilters. Both functors can be regarded as subfunctors of the contravariant powerset functor $\mathcal{2}$, as they both map a morphism f in their respective categories to the inverse image function f^{-1}. Composing these functors, we find that for a set X, $\mathbb{UP}(X) = \mathrm{Uf}(X)$, and for a function $f\colon X \to Y$, $\mathbb{UP}(f) = (f^{-1})^{-1}$. Hence Uf can be regarded as a subfunctor of $\mathcal{2}^{\mathcal{2}}$.

The following definition of ultrafilter extensions of neighbourhood models is obtained by instantiating the corresponding coalgebraic notion for F-coalgebras in [89] to the case that $F = \mathcal{2}^{\mathcal{2}}$. We sketch the main ideas of the construction in Remark 5.4.20 below. In fact, the definition of the neighbourhood relation of the ultrafilter extension goes back to the definition of the canonical neighbourhood model in [141].

5.4.19. Definition. *Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model. The* ultrafilter extension *of $\mathcal{M}$ is defined as the triple $\mathcal{M}^u := \langle \mathrm{Uf}(S), \mu, V^u\rangle$, where*

- $\mathrm{Uf}(S)$ *is the set of ultrafilters over the set S,*

- $\mu : \mathrm{Uf}(S) \to \mathcal{2}^{\mathcal{2}}(\mathrm{Uf}(S))$ *is defined by*

$$\mu(\mathbf{u}) := \{\hat{U} \subseteq \mathrm{Uf}(S) \mid U \subseteq S,\ \boxtimes U \in \mathbf{u}\},$$

  where for any $U \subseteq S$ *we put* $\boxtimes U := \{s \in S \mid U \in \nu(s)\}$,

- $V^u(p) := \{\mathbf{u} \in \mathrm{Uf}(S) \mid V(p) \in \mathbf{u}\}$. ◁

5.4.20. Remark. In [89], a general construction of ultrafilter extensions of F-coalgebras is given. We now sketch how this framework instantiates to $\mathcal{2}^{\mathcal{2}}$-coalgebras, i.e., neighbourhood frames.

Classical modal logic can be described as a functor $\mathbb{L}\colon \mathsf{BA} \to \mathsf{BA}$. For a Boolean algebra $\mathcal{A} = \langle A, \wedge, \vee, -0, 1\rangle$, $\mathbb{L}(\mathcal{A})$ is the free Boolean algebra generated by $\{\Box a \mid a \in A\}$. Let $\mathsf{Alg}(L)$ be the category of $\mathbb{L}$-algebras over BA. The functors $\mathbb{P}\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{BA}$ and $\mathbb{U}\colon \mathsf{BA} \to \mathsf{Set}^{\mathrm{op}}$ are extended to functors $\bar{\mathbb{P}}\colon \mathsf{Coalg}(\mathcal{2}^{\mathcal{2}})^{\mathrm{op}} \to \mathsf{Alg}(\mathbb{L})$ and $\bar{\mathbb{U}}\colon \mathsf{Alg}(\mathbb{L}) \to \mathsf{Coalg}(\mathcal{2}^{\mathcal{2}})^{\mathrm{op}}$. The ultrafilter extension of a $\mathcal{2}^{\mathcal{2}}$-coalgebra $\langle S, \nu\rangle$ is then obtained as $\bar{\mathbb{U}}\bar{\mathbb{P}}(\langle S, \nu\rangle)$. The lifting of $\mathbb{P}$ and $\mathbb{U}$ relies on the existence of two natural transformations: $\delta\colon \mathbb{L}\mathbb{P} \to \mathbb{P}\mathcal{2}^{\mathcal{2}}$ and $h\colon \mathbb{U}\mathbb{L} \to \mathcal{2}^{\mathcal{2}}\mathbb{U}$. For a set X, δ_X and h_X are given by (cf. Definition 2.6.5 and Example 3.6 of [89]):

$$\begin{array}{rrcl} \delta_X\colon & \mathbb{L}\mathbb{P}(X) & \to & \mathbb{P}\mathcal{2}^{\mathcal{2}}(X) \\ & \Box U & \mapsto & \{N \in \mathcal{2}^{\mathcal{2}}(X) \mid U \in N\} \\ h_X\colon & \mathbb{U}\mathbb{L}(X) & \to & \mathcal{2}^{\mathcal{2}}\mathbb{U}(X) \\ & \mathbf{u} & \mapsto & \{\hat{U} \subseteq \mathbb{U}\mathbb{P}(X) \mid \Box U \in U\} \end{array}$$

The liftings $\bar{\mathbb{P}}$ and $\bar{\mathbb{U}}$ are now given as follows on objects: $\bar{\mathbb{P}}$ maps a $\mathcal{2}^{\mathcal{2}}$-coalgebra $\langle X, \nu\rangle$ to $\bar{\mathbb{P}}(\langle X, \nu\rangle) = \langle \mathbb{L}\mathbb{P}(X), \mathbb{P}(\nu) \circ \delta_X\rangle$ as illustrated here:

$$\mathbb{L}\mathbb{P}(X) \xrightarrow{\delta_X} \mathbb{P}\mathcal{2}^{\mathcal{2}}(X) \xrightarrow{\mathbb{P}(\nu)} \mathbb{P}(X)$$

$\bar{\mathbb{U}}$ maps a $\langle \mathcal{A}, \alpha\rangle$ in $\mathsf{Alg}(\mathbb{L})$ to $\bar{\mathbb{U}}(\langle \mathcal{A}, \alpha\rangle) = \langle \mathbb{U}(\mathcal{A}), h_{\mathcal{A}} \circ \mathbb{U}(\alpha)\rangle$:

$$\mathbb{U}(\mathcal{A}) \xrightarrow{\mathbb{U}(\alpha)} \mathbb{U}\mathbb{L}(\mathcal{A}) \xrightarrow{h_{\mathcal{A}}} \mathcal{2}^{\mathcal{2}}(\mathbb{U}(\mathcal{A}))$$

By working out the details, the reader can now confirm that the composition $\bar{\mathbb{U}}\bar{\mathbb{P}}$ yields the ultrafilter extension of neighbourhood frames provided in Definition 5.4.19. ◁

The construction of the ultrafilter extension in Definition 5.4.19 can be seen as an extension of the Set-functor $\mathrm{Uf}\colon \mathsf{Set} \to \mathsf{Set}$ to a functor $(\_)^u\colon \mathsf{Nbhd} \to \mathsf{Nbhd}$ such that for any neighbourhood model $\mathcal{M}$, the principal ultrafilter map u is truth-preserving injective map from $\mathcal{M}$ into $\mathcal{M}^u$. In order to see that the construction $(\_)^u$ of the ultrafilter extension is functorial we show that bounded morphisms between neighbourhood models induce bounded morphisms between the corresponding ultrafilter extensions.

5.4.21. LEMMA. *Let $\mathcal{M}_1 = \langle S_1, \nu_1, V_1\rangle$ and $\mathcal{M}_2 = \langle S_2, \nu_2, V_2\rangle$ be neighbourhood models an let $f : S_1 \to S_2$ be a bounded morphism from $\mathcal{M}_1$ to $\mathcal{M}_2$. The function $f^u := \mathrm{Uf}(f)$ is a bounded morphism from $\mathcal{M}_1^u = \langle \mathrm{Uf}(S_1), \mu_1, V_1^u\rangle$ to $\mathcal{M}_2^u = \langle \mathrm{Uf}(S_2), \mu_2, V_2^u\rangle$.*

PROOF. First observe that for any subset $U \subseteq S_2$:

$$(f^u)^{-1}[\hat{U}] = \widehat{f^{-1}[U]}, \tag{5.11}$$

since for any $\mathbf{u} \in \mathrm{Uf}(S_1)$: $\mathbf{u} \in (f^u)^{-1}[\hat{U}]$ iff $f^u(\mathbf{u}) \in \hat{U}$ iff $U \in f^u(\mathbf{u}) = \mathcal{2}^f(\mathbf{u})$ iff $f^{-1}[U] \in \mathbf{u}$ iff $\mathbf{u} \in \widehat{f^{-1}[U]}$. Moreover,

$$f^{-1}[\boxtimes U] = \boxtimes(f^{-1}[U]) \tag{5.12}$$

since for any $s \in S_1$ and $U \subseteq S_2$: $s \in f^{-1}[\boxtimes U]$ iff $f(s) \in \boxtimes U$ iff $U \in \nu_2(f(s))$ iff $f^{-1}[U] \in \nu_1(s)$ iff $s \in \boxtimes(f^{-1}[U])$.

To prove that f^u is a bounded morphism, let $\mathbf{u} \in \mathrm{Uf}(S_1)$ and $U \subseteq S_2$. We now have:

$$\begin{aligned}
\hat{U} \in \mu_2(f^u(\mathbf{u})) \quad &\text{iff} \quad \boxtimes U \in f^u(\mathbf{u}) = \mathcal{2}^f(\mathbf{u}) \\
&\text{iff} \quad f^{-1}[\boxtimes U] \overset{(5.12)}{=} \boxtimes(f^{-1}[U]) \in \mathbf{u} \\
&\text{iff} \quad \widehat{f^{-1}[U]} \overset{(5.11)}{=} (f^u)^{-1}[\hat{U}] \in \mu_1(\mathbf{u}).
\end{aligned}$$

It is easily verified that f^u respects valuations: $V_1(p) \in \mathbf{u}$ iff $f^{-1}[V_2(p)] \in \mathbf{u}$ iff $V_2(p) \in f^u(\mathbf{u})$. QED

The next proposition connects truth of a modal formula in the ultrafilter extension to the truth set of the formula in the original model.

5.4.22. PROPOSITION. *Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model with ultrafilter extension $\mathcal{M}^u$. For all $\mathbf{u} \in \mathrm{Uf}(S)$ and for all formulas $\varphi \in \mathcal{L}$ we have*

$$\mathcal{M}^u, \mathbf{u} \models \varphi \quad \textit{iff} \quad [\![\varphi]\!]^{\mathcal{M}} \in \mathbf{u}.$$

PROOF. The standard proof is obtained by induction on the formula φ. We only treat the case of the modal operator in detail: Suppose φ is of the form $\Box\psi$ and let $\mathbf{u} \in \mathrm{Uf}(S)$ be an ultrafilter. Then

$$\begin{aligned}
\mathcal{M}^u, \mathbf{u} \models \Box\psi \quad &\text{iff} \quad [\![\psi]\!]^{\mathcal{M}^u} \overset{(\text{I.H.})}{=} \{\mathbf{v} \in \mathrm{Uf}(S) \mid [\![\psi]\!]^{\mathcal{M}} \in \mathbf{v}\} \in \mu(\mathbf{u}) \\
&\text{iff} \quad \widehat{[\![\psi]\!]^{\mathcal{M}}} \in \mu(\mathbf{u}) \\
&\text{iff} \quad \boxtimes([\![\psi]\!]^{\mathcal{M}}) = [\![\Box\psi]\!]^{\mathcal{M}} \in \mathbf{u}.
\end{aligned}$$

QED

Using Proposition 5.4.22, we now easily show that the principal ultrafilter map u preserves the truth of modal formulas. However, it is important to note that, in general, u is not a bounded morphism from a model $\mathcal{M} = \langle S, \nu, V\rangle$ to its ultrafilter extension $\mathcal{M}^u$.

5.4.23. Lemma. *Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model with ultrafilter extension $\mathcal{M}^u = \langle \mathrm{Uf}(S), \mu, V^u\rangle$ and let $\mathsf{u} : S \to \mathrm{Uf}(S)$ be the injective map from S to $\mathrm{Uf}(S)$. For every modal formula φ we have $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}^u, \mathsf{u}_s \models \varphi$.*

Proof. Let $s \in S$ and let φ be modal formula. Then $\mathcal{M}, s \models \varphi$ iff $s \in [\![\varphi]\!]^{\mathcal{M}}$ iff $[\![\varphi]\!]^{\mathcal{M}} \in \mathsf{u}_s$ iff $\mathcal{M}^u, \mathsf{u}_s \models \varphi$ where the last equivalence is a consequence of Proposition 5.4.22. QED

Another consequence of Proposition 5.4.22 is the fact that ultrafilter extensions are modally saturated.

5.4.24. Proposition. *For any neighbourhood model $\mathcal{M}$, the ultrafilter extension $\mathcal{M}^u$ is modally saturated.*

Proof. Let $\mathcal{M} = \langle S, \nu, V\rangle$ and $\mathcal{M}^u = \langle \mathrm{Uf}(S), \mu, V^u\rangle$. We show that any $\hat{U} \subseteq \mathrm{Uf}(S)$ is compact. This suffices since all neighbourhoods in $\mathcal{M}^u$ are of the form $\hat{U} \subseteq \mathrm{Uf}(S)$ and for any $\hat{U}$, $\mathrm{Uf}(S) \setminus \hat{U} = \widehat{U^c}$. Let Ψ be a set of formulas with the property that Ψ is finitely satisfiable in $\hat{U}$. For any finite set of formulas $\{\psi_1, \ldots, \psi_n\} \subseteq \Psi$ there exists therefore an ultrafilter $\mathbf{u} \in \hat{U}$ such that $\mathcal{M}^u, \mathbf{u} \models \psi_1 \wedge \ldots \wedge \psi_n$. This implies by Proposition 5.4.22 that

$$\{[\![\psi_1]\!]^{\mathcal{M}}, \ldots, [\![\psi_n]\!]^{\mathcal{M}}\} \cup \{U\} \subseteq \mathbf{u}$$

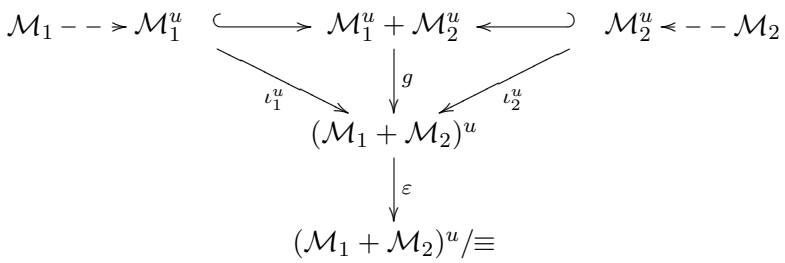
Since $\mathbf{u}$ is closed under finite intersections this implies $[\![\psi_1]\!]^{\mathcal{M}} \cap \ldots \cap [\![\psi_n]\!]^{\mathcal{M}} \cap U \in \mathbf{u}$ and hence $[\![\psi_1]\!]^{\mathcal{M}} \cap \ldots \cap [\![\psi_n]\!]^{\mathcal{M}} \cap U \neq \emptyset$. As the set $\{\psi_1, \ldots, \psi_n\}$ was arbitrary we conclude that the set $X := \{U\} \cup \{[\![\psi]\!]^{\mathcal{M}} \mid \psi \in \Psi\}$ has the finite intersection property. Hence by the ultrafilter theorem, there exists some ultrafilter $\mathbf{u}' \in \mathrm{Uf}(S)$ such that $X \subseteq \mathbf{u}'$. By construction we get $\mathbf{u}' \in \hat{U}$ and again by Proposition 5.4.22, that Ψ is satisfiable at $\mathbf{u}' \in \hat{U}$. QED

We are now able to prove that the class of ultrafilter extensions of neighbourhood models is a Hennessy-Milner class.

5.4.25. Proposition. *The class $\mathbf{U} := \{\mathcal{M}^u \mid \mathcal{M} \in \mathsf{Nbhd}\}$ of ultrafilter extensions of neighbourhood models is a Hennessy-Milner class.*

Proof. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be arbitrary neighbourhood models. By Lemma 5.4.2 it suffices to show that modal equivalence is a congruence on the disjoint union $\mathcal{M}_1^u + \mathcal{M}_2^u$ of their ultrafilter extensions. By Proposition 5.4.24, $(\mathcal{M}_1 + \mathcal{M}_2)^u$ is

modally saturated, hence the quotient map $\varepsilon\colon (\mathcal{M}_1+\mathcal{M}_2)^u \to (\mathcal{M}_1+\mathcal{M}_2)^u/{\equiv}$ is a bounded morphism. Furthermore, denote by $\iota_i\colon \mathcal{M}_i \to \mathcal{M}_1+\mathcal{M}_2$, $i\in\{1,2\}$, the canonical inclusion morphisms. By Lemma 5.4.21, $\iota_i^u\colon \mathcal{M}_i^u \to (\mathcal{M}_1+\mathcal{M}_2)^u$, $i\in\{1,2\}$, are bounded morphisms, hence there exists, by the universal property of the disjoint union $\mathcal{M}_1^u+\mathcal{M}_2^u$, a bounded morphism g such that the following diagram commutes:



Hence $\varepsilon\circ g\colon \mathcal{M}_1^u+\mathcal{M}_2^u \to (\mathcal{M}_1+\mathcal{M}_2)^u/{\equiv}$ is a bounded morphism, and two ultrafilters in $\mathcal{M}_1^u+\mathcal{M}_2^u$ are modally equivalent if and only if they are identified by $\varepsilon\circ g$. It follows that on $\mathcal{M}_1^u+\mathcal{M}_2^u$, the modal equivalence relation is the kernel of $\varepsilon\circ g$, and hence a congruence. QED

As a corollary we obtain the behavioural-equivalence-somewhere-else result.

5.4.26. Theorem. *Let $\mathcal{M}_1=\langle S_1,\nu_1,V_1\rangle$ and $\mathcal{M}_2=\langle S_2,\nu_2,V_2\rangle$ be neighbourhood models with the respective ultrafilter extensions $\mathcal{M}_1^u$ and $\mathcal{M}_2^u$. For all states $s_1\in S_1$ and $s_2\in S_2$ we have*

$$\mathcal{M}_1,s_1\equiv\mathcal{M}_2,s_2 \quad\Rightarrow\quad \mathcal{M}_1^u,\mathsf{u}_{s_1}\sim_b\mathcal{M}_2^u,\mathsf{u}_{s_2}.$$

Proof. Let s_1 and s_2 be modally equivalent states in $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. By Lemma 5.4.23 the states u_{s_1} and u_{s_2} of the ultrafilter extensions $\mathcal{M}_1^u$ and $\mathcal{M}_2^u$ are modally equivalent as well. The claim is now a direct consequence of Prop 5.4.25. QED

5.5 Model-theoretic results

5.5.1 The classical modal fragment of first-order logic

We will now prove that the three equivalence notions described in section 5.3 all characterise the modal fragment of first-order logic over the class of neighbourhood models (Theorem 5.5.5). This result is an analogue of Van Benthem's characterisation theorem for normal modal logic (cf. [20]): *On the class of Kripke models, modal logic is the Kripke bisimulation-invariant fragment of first-order*

logic. It is well-known that, when interpreted over Kripke models, the basic modal language $\mathcal{L}$ can be seen as a fragment of a first-order language which has a binary predicate $\mathsf{R}_\Box$, and a unary predicate P for each atomic proposition p in the modal language. Formulas of this first-order language can be interpreted in Kripke models in the obvious way. Van Benthem's theorem tells us that a first-order formula $\alpha(x)$ is invariant under Kripke bisimulation if and only if $\alpha(x)$ is equivalent to a modal formula.

The first step towards a Van Benthem-style characterisation theorem for classical modal logic is to show how $\mathcal{L}$ can be viewed as a fragment of first-order logic. We will translate modal formulas into a two-sorted first-order language $\mathcal{L}_1$, which has previously been employed in proving a Van Benthem style characterisation theorems for topological modal logic [32] and monotonic modal logic [113], and for reasoning about topological models more generally [42]. In Remark 5.5.8) we will give a more detailed comparison between our characterisation theorem and the characterisation theorem for monotonic modal logic given in [113]. The two sorts of the language $\mathcal{L}_1$ are denoted s and n. Terms of sort s are intended to represent states, whereas terms of sort n are intended to represent neighbourhoods. We assume there are countable sets of variables of each sort. To simplify notation, we will not state the type of variables explicitly. Instead we use the following conventions: $x, y, x', y', x_1, y_2, \ldots$ denote variables of sort s (*state variables*) and $u, v, u', v', u_1, v_1, \ldots$ denote variables of sort n (*neighbourhood variables*). Furthermore, the language $\mathcal{L}_1$ contains a unary predicate P_i (of sort s) for each $i \in \omega$, a binary relation symbol N relating elements of sort s to elements of sort n, and a binary relation symbol E relating elements of sort n to elements of sort s. The intended interpretation of $x\mathsf{N}u$ is "u is a neighbourhood of x", and the intended interpretation of $u\mathsf{E}x$ is "x is an element of u". The language $\mathcal{L}_1$ is generated by the following grammar:

$$\varphi, \psi \quad ::= \quad x = y \mid u = v \mid \mathsf{P}_i x \mid x\mathsf{N}u \mid u\mathsf{E}x \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x\varphi \mid \exists u\varphi$$

where $i \in \omega$; x and y are state variables of sort s; and u and v are neighbourhood variables of sort n. The usual abbreviations (eg. $\forall$ for $\neg\exists\neg$) apply.

Formulas of $\mathcal{L}_1$ are interpreted in two-sorted first-order structures of the type $\mathfrak{M} = \langle D^{\mathsf{s}}, D^{\mathsf{n}}, \{P_i \mid i \in \omega\}, N, E\rangle$ where D^{s} and D^{n} are the carrier sets of sort s and sort n, respectively, and each $P_i \subseteq D^{\mathsf{s}}$, $N \subseteq D^{\mathsf{s}} \times D^{\mathsf{n}}$ and $E \subseteq D^{\mathsf{n}} \times D^{\mathsf{s}}$. The usual definitions of free and bound variables apply. Truth of sentences (formulas with no free variables) $\varphi \in \mathcal{L}_1$ in a structure $\mathfrak{M}$ (denoted $\mathfrak{M} \models \varphi$) is defined as expected. If x is a free state variable in φ (denoted $\varphi(x)$), then we write $\mathfrak{M} \models \varphi[s]$ to mean that φ is true in $\mathfrak{M}$ when $s \in D^{\mathsf{s}}$ is assigned to x. Note that $\mathfrak{M} \models \exists x\varphi$ iff there is an element $s \in D^{\mathsf{s}}$ such that $\mathfrak{M} \models \varphi[s]$. If Ψ is a set of $\mathcal{L}_1$-formulas, and $\mathfrak{M}$ is an $\mathcal{L}_1$-model, then $\mathfrak{M} \models \Psi$ means that for all $\psi \in \Psi$, $\mathfrak{M} \models \psi$. Given a class $\mathbf{K}$ of $\mathcal{L}_1$-models, we denote the *semantic consequence relation over* $\mathbf{K}$ by $\models_{\mathbf{K}}$. In particular, for $\Psi(x) \cup \{\varphi(x)\} \subseteq \mathcal{L}_1$, $\Psi(x) \models_{\mathbf{K}} \varphi(x)$ if

for all $\mathfrak{M} \in \mathbf{K}$ and all s of sort s in $\mathfrak{M}$, $\mathfrak{M} \models \Phi[s]$ implies $\mathfrak{M} \models \varphi[s]$. Moreover, a set of formulas $\Phi(x)$ is $\mathbf{K}$-consistent ($\Phi(x) \not\models_{\mathbf{K}} \bot$) if there exists an $\mathfrak{M} \in \mathbf{K}$ and an s of sort s in $\mathfrak{M}$ such that $\mathfrak{M} \models \Phi[s]$.

We can now translate modal $\mathcal{L}$-formulas and neighbourhood models to the first-order setting in a natural way:

5.5.1. Definition. Let $\mathcal{M} = \langle S, \nu, V\rangle$ be a neighbourhood model. The *first-order translation* of $\mathcal{M}$ is the structure $\mathcal{M}^\circ = \langle D^{\mathsf{s}}, D^{\mathsf{n}}, \{P_i \mid i \in \omega\}, R_\nu, R_\ni\rangle$ where

$$\begin{array}{lcl} D^{\mathsf{s}} & = & S, \\ D^{\mathsf{n}} & = & \nu[S] = \bigcup_{s\in S} \nu(s) \\ P_i & = & V(p_i) \text{ for each } i \in \omega, \\ R_\nu & = & \{\langle s, U\rangle \mid s \in D^{\mathsf{s}}, U \in \nu(s)\}, \\ R_\ni & = & \{\langle U, s\rangle \mid s \in D^{\mathsf{s}}, s \in U\}. \end{array}$$ ◁

5.5.2. Definition. The *standard translation* of the basic modal language is a family of functions $st_x : \mathcal{L} \to \mathcal{L}_1$ defined as follows: $st_x(\bot) = \neg(x = x)$, $st_x(p_i) = \mathsf{P}_i x$, $st_x(\neg\varphi) = \neg st_x(\varphi)$, $st_x(\varphi \wedge \psi) = st_x(\varphi) \wedge st_x(\psi)$, and

$$st_x(\Box\varphi) = \exists u(x\mathsf{N}u \wedge (\forall y(u\mathsf{E}y \leftrightarrow st_y(\varphi))).$$ ◁

This translation preserves truth; the easy proof is left to the reader.

5.5.3. Lemma. *Let $\mathcal{M}$ be a neighbourhood model and $\varphi \in \mathcal{L}$. For each $s \in S$, $\mathcal{M}, s \models \varphi$ iff $\mathcal{M}^\circ \models st_x(\varphi)[s]$.*

In the Kripke case, every first-order model for the language with $\mathsf{R}_\Box$ can be seen as Kripke model. However, it is not the case that every $\mathcal{L}_1$-structure is the translation of a neighbourhood model. Luckily, we can axiomatize the subclass of neighbourhood models up to isomorphism. Let NAX be the following axioms

(A1) $\forall u \exists x(x\mathsf{N}u)$

(A2) $\forall u, v((\forall x(u\mathsf{E}x \leftrightarrow v\mathsf{E}x)) \to u = v)$

It is not hard to see that if $\mathcal{M}$ is a neighbourhood model, then $\mathcal{M}^\circ \models \mathsf{NAX}$. The next result states that, in fact, NAX completely characterises the class $\mathbf{N} := \{\mathfrak{M} \mid \mathfrak{M} \cong \mathcal{M}^\circ \text{ for some neighbourhood model } \mathcal{M}\}$, where $\cong$ denotes isomorphism of $\mathcal{L}_1$-models.

5.5.4. Proposition. *Suppose $\mathfrak{M}$ is an $\mathcal{L}_1$-model and $\mathfrak{M} \models \mathsf{NAX}$. Then there is a neighbourhood model $\mathfrak{M}_\circ$ such that $\mathfrak{M} \cong (\mathfrak{M}_\circ)^\circ$.*

PROOF. Let $\mathfrak{M} = \langle D^{\mathsf{s}}, D^{\mathsf{n}}, \{P_i \mid i \in \omega\}, N, E\rangle$ be an $\mathcal{L}_1$-model such that $\mathfrak{M} \models \mathsf{NAX}$. We will construct from $\mathfrak{M}$ a neighbourhood model $\mathfrak{M}_\circ = \langle S, \nu, V\rangle$ such that $\mathfrak{M} \cong (\mathfrak{M}_\circ)^\circ$. In case $D^{\mathsf{s}} = \emptyset$ we also have $D^{\mathsf{n}} = \emptyset$ by axiom $\mathsf{A1}$ and hence we define $\mathfrak{M}_\circ$ to be the empty neighbourhood model. In the case $D^{\mathsf{s}} \neq \emptyset$ we first define a map $\eta : D^{\mathsf{n}} \to \mathcal{P}(D^{\mathsf{s}})$ by $\eta(u) = \{s \in D^{\mathsf{s}} \mid uEs\}$. We take $S = D^{\mathsf{s}}$. Now define for each $s \in S$ and each $X \subseteq S$: $X \in \nu(s)$ iff there is a $u \in D^{\mathsf{n}}$ such that sNu and $X = \eta(u)$, and define for all $i \in \omega$, $V(p_i) = \{s \in S \mid \mathfrak{M} \models \mathsf{P}_i[s]\}$. Then $\mathfrak{M}_\circ$ is clearly a well-defined neighbourhood model, and it is not hard to see that the maps $id : D^{\mathsf{s}} \to D^{\mathsf{s}}$ and $\eta : D^{\mathsf{n}} \to \bigcup_{s\in D^{\mathsf{s}}} \nu(s)$ yield an isomorphism from $\mathfrak{M}$ to $(\mathfrak{M}_\circ)^\circ = \langle S, \nu[S], \{P_i' \mid i \in \omega\}, R_\nu, R_\ni\rangle$ (cf. Definition 5.5.1). The details are left to the reader. QED

Thus, in a precise way, we can think of models in $\mathbf{N}$ as neighbourhood models. In particular, if $\mathfrak{M}$ and $\mathfrak{N}$ are in $\mathbf{N}$ we will write $\mathfrak{M} + \mathfrak{N}$ by which we (strictly speaking) mean the $\mathcal{L}_1$-model $(\mathfrak{M}_\circ + \mathfrak{N}_\circ)^\circ$ (which is also in $\mathbf{N}$). Furthermore, Proposition 5.5.4 implies that we can work relative to $\mathbf{N}$ while still preserving nice first-order properties such as compactness and the existence of countably saturated models. These properties are essential in the proof of Theorem 5.5.5.

5.5.2 Characterisation theorem

We are now able to formulate our characterisation theorem. Let $\sim$ be a relation on model-state pairs. Over the class $\mathbf{N}$, an $\mathcal{L}_1$-formula $\alpha(x)$ is *invariant under* $\sim$, if for all models $\mathfrak{M}_1$ and $\mathfrak{M}_2$ in $\mathbf{N}$ and all sort s-domain elements s_1 and s_2 of $\mathfrak{M}_1$ and $\mathfrak{M}_2$, respectively, we have $\mathfrak{M}_1, s_1 \sim \mathfrak{M}_2, s_2$ implies $\mathfrak{M}_1 \models \alpha[s_1]$ iff $\mathfrak{M}_2 \models \alpha[s_2]$. Over the class $\mathbf{N}$, an $\mathcal{L}_1$-formula $\alpha(x)$ *is equivalent to the translation of a modal formula* if there is a modal formula $\varphi \in \mathcal{L}$ such that for all models $\mathfrak{M}$ in $\mathbf{N}$, and all s-domain elements s in $\mathfrak{M}$, $\mathfrak{M} \models \alpha[s]$ iff $\mathfrak{M} \models st_x(\varphi)[s]$.

5.5.5. THEOREM (CHARACTERISATION). *Let $\alpha(x)$ be an $\mathcal{L}_1$-formula. Over the class $\mathbf{N}$ the following are equivalent:*

1. *$\alpha(x)$ is equivalent to the translation of a modal formula,*
2. *$\alpha(x)$ is invariant under behavioural equivalence,*
3. *$\alpha(x)$ is invariant under precocongruences,*
4. *$\alpha(x)$ is invariant under 2^2-bisimilarity.*

Our proof of Theorem 5.5.5 uses essentially the same ingredients as the proof of Van Benthem's theorem (see e.g. [25]) where the main steps are:

1. Given a Kripke model $\mathcal{M}$ we can obtain a modally saturated, elementary extension $\mathcal{M}^*$ of $\mathcal{M}$.

2. Between modally saturated Kripke models, modal equivalence is a Kripke bisimulation.

Together, 1 and 2 imply that modally equivalent states $\mathcal{M}, s$ and $\mathcal{N}, t$ are Kripke bisimilar in their modally saturated, elementary extensions $\mathcal{M}^*, s^*$ and $\mathcal{N}^*, t^*$. Our analogue of 2 is that in a modally saturated neighbourhood model, modal equivalence is a congruence, which we have shown in Proposition 5.4.6. If we can show an analogue of 1, it follows that if $\mathcal{M}, s$ and $\mathcal{N}, t$ are modally equivalent, then they have behaviourally equivalent representatives in a modally saturated, elementary extension of $\mathcal{M} + \mathcal{N}$.

As in the Kripke case, we can obtain an ω-saturated, elementary extension of any $\mathcal{L}_1$-model in the form of an ultrapower using standard first-order logic techniques (see e.g. [34]). It then only remains to show that an ω-saturated neighbourhood model (viewed as a $\mathcal{L}_1$-model) is modally saturated. Before we state and prove this lemma, we recall (cf. [34]) the definition of ω-saturation. Let $\mathfrak{M}$ be a first-order $\mathcal{L}_1$-model with domain M. For a subset $C \subseteq M$, the *C-expansion $\mathcal{L}_1[C]$ of $\mathcal{L}_1$* is the two-sorted first-order language obtained from $\mathcal{L}_1$ by adding a constant $\underline{c}$ for each $c \in C$. Now $\mathcal{L}_1[C]$-formulas are interpreted in $\mathfrak{M}$ by requiring that a new constant $\underline{c}$ is interpreted as the element c. The $\mathcal{L}_1$-model $\mathfrak{M}$ is *ω-saturated*, if for every finite $C \subseteq_\omega M$, and every collection $\Gamma(x)$ of $\mathcal{L}_1[C]$-formulas with one free variable x the following holds: If $\Gamma(x)$ is finitely satisfiable in $\mathfrak{M}$ (equivalently, if $\Gamma(x)$ is consistent with the $\mathcal{L}_1[C]$ theory of $\mathfrak{M}$), then $\Gamma(x)$ is satisfiable in $\mathfrak{M}$. It is a classic result of model theory that every model has an ω-saturated elementary extension (cf. [34]).

5.5.6. Lemma. *Let $\mathfrak{M}$ be a model in $\mathbf{N}$, and let $\mathfrak{M}_\circ$ be its corresponding neighbourhood model. If $\mathfrak{M}$ is ω-saturated, then $\mathfrak{M}_\circ$ is modally saturated.*

Proof. Let $\mathfrak{M}$ be an $\mathcal{L}_1$-model in $\mathbf{N}$, $\mathfrak{M}_\circ = \langle S, \nu, V \rangle$ its corresponding neighbourhood model (cf. Proposition 5.5.4), and assume that $\mathfrak{M}$ is ω-saturated. Let Ψ be a set of modal $\mathcal{L}$-formulas, and let $U \subseteq S$ be a neighbourhood of some state s. Then U corresponds to a domain element $u \in D^{\mathsf{n}}$ of $\mathfrak{M}$ via the isomorphism $\mathfrak{M} \cong (\mathfrak{M}_\circ)^\circ$. If Ψ is finitely satisfiable in U in $\mathfrak{M}_\circ$, then the set of $\mathcal{L}_1[\{u\}]$-formulas $\{\underline{u}\mathsf{N}x\} \cup \{st_x(\psi) \mid \psi \in \Psi\}$ is finitely satisfiable in $\mathfrak{M}$, and hence satisfiable, which implies that Ψ is satisfiable in U. Similarly, if Ψ is finitely satisfiable in U^c, then the set of $\mathcal{L}_1[\{u\}]$-formulas $\{\neg\underline{u}\mathsf{N}x\} \cup \{st_x(\psi) \mid \psi \in \Psi\}$ is finitely satisfiable in $\mathfrak{M}$, and hence satisfiable, which implies that Ψ is satisfiable in U^c. QED

We are now ready to prove Theorem 5.5.5.

PROOF OF THEOREM 5.5.5. It is clear that *2* $\Rightarrow$ *3* $\Rightarrow$ *4* (cf. Proposition 5.3.8). To see that *4* $\Rightarrow$ *2*, we only need to recall (cf. [129]) that graphs of bounded morphisms are 2^2-bisimulations. Furthermore, as truth of modal formulas is preserved by behavioural equivalence, *1* $\Rightarrow$ *2* is clear. We complete the proof by showing that *2* $\Rightarrow$ *1*.

Let $\mathrm{MOC}_{\mathbf{N}}(\alpha) = \{st_x(\varphi) \mid \varphi \in \mathcal{L}, \alpha(x) \models_{\mathbf{N}} st_x(\varphi)\}$ be the set of modal consequences of $\alpha(x)$ over the class $\mathbf{N}$. It suffices to show that $\mathrm{MOC}_{\mathbf{N}}(\alpha) \models_{\mathbf{N}} \alpha(x)$, since then by compactness there is a finite subset $\Gamma(x) \subseteq \mathrm{MOC}_{\mathbf{N}}(\alpha)$ such that $\Gamma(x) \models_{\mathbf{N}} \alpha(x)$ and $\alpha(x) \models_{\mathbf{N}} \bigwedge\Gamma(x)$. It follows that over $\mathbf{N}$, $\alpha(x)$ is equivalent to $\bigwedge\Gamma(x)$, which is the translation of a modal formula. So suppose $\mathfrak{M}$ is a model in $\mathbf{N}$ and $\mathrm{MOC}_{\mathbf{N}}(\alpha)$ is satisfied at some element s in $\mathfrak{M}$. We must show that $\mathfrak{M} \models \alpha[s]$. Consider the set $T(x) = \{st_x(\varphi) \mid \mathfrak{M}_\circ, s \models \varphi\} \cup \{\alpha(x)\}$. $T(x)$ is $\mathbf{N}$-consistent, since suppose to the contrary that $T(x)$ is $\mathbf{N}$-inconsistent, then by compactness, there is a finite collection of modal formulas $\varphi_1, \ldots, \varphi_n$ such that $\mathfrak{M}_\circ, s \models \varphi_i$ for all $i = 1, \ldots, n$ and $\alpha(x) \models_{\mathbf{N}} \neg\bigwedge_{i=1}^n st_x(\varphi_i)$, which implies that $\neg\bigwedge_{i=1}^n st_x(\varphi_i) \in \mathrm{MOC}_{\mathbf{N}}(\alpha)$. But this contradicts the assumption that $\mathfrak{M} \models \mathrm{MOC}_{\mathbf{N}}(\alpha)[s]$ and $\mathfrak{M} \models st_x(\varphi_i)[s]$ for all $i = 1, \ldots, n$. Hence $T(x)$ is satisfied at an element t in some $\mathfrak{N} \in \mathbf{N}$, and by construction, s and t are modally equivalent: For all modal formulas $\varphi \in \mathcal{L}$, $\mathfrak{M} \models st_x(\varphi)[s]$ implies $st_x(\varphi) \in T(x)$, and hence $\mathfrak{N} \models st_x(\varphi)[t]$. Conversely, $\mathfrak{M} \not\models st_x(\varphi)[s]$ iff $\mathfrak{M} \models \neg st_x(\varphi)[s]$ which implies $st_x(\neg\varphi) = \neg st_x(\varphi) \in T(x)$, and hence $\mathfrak{N} \not\models st_x(\varphi)[t]$.

Take now an ω-saturated, elementary extension $\mathfrak{U}$ of $\mathfrak{M} + \mathfrak{N}$. Note that $\mathfrak{U} \in \mathbf{N}$, since validity of NAX is preserved under elementary extensions. Moreover, the images s_U and t_U in $\mathfrak{U}$ of s and t, respectively, are also modally equivalent, since modal truth is transferred by elementary maps. Now since $\mathfrak{U}$ is ω-saturated and thus by Lemma 5.5.6, $\mathfrak{U}_\circ$ is modally saturated, it follows from Proposition 5.4.6 that s_U and t_U are behaviourally equivalent. The construction is illustrated in the following diagram; $\preceq$ indicates that the map is elementary, and ι and κ are the canonical inclusions.

$$\begin{array}{ccccccc} \mathrm{MOC}_{\mathbf{N}}(\alpha)[s] & \rightleftharpoons & \mathfrak{M} & \xrightarrow{\iota} & \mathfrak{M} + \mathfrak{N} & \xleftarrow{\kappa} & \mathfrak{N} \models \alpha[t] \\ & & & & \downarrow \preceq & & \\ & & & & \mathfrak{U} & & \end{array}$$

Finally, we can transfer the truth of $\alpha(x)$ from $\mathfrak{N}, t$ to $\mathfrak{M}, s$ by using the invariance of modal formulas under bounded morphisms and standard translations *(bm+st)*; elementary maps *(elem)*; and the assumption that $\alpha(x)$ is invariant under behavioural equivalence *($\alpha(x)$-beh-inv)*.

$$\begin{array}{lll}
\mathfrak{N} \models \alpha[t] & \iff (\mathfrak{M}_\circ + \mathfrak{N}_\circ)^\circ \models \alpha[\kappa(t)] & \textit{(bm+st)} \\
& \iff \mathfrak{U} \models \alpha[t_U] & \textit{(elem)} \\
& \iff \mathfrak{U} \models \alpha[s_U] & (s_U \sim_b t_U \textit{ and } \alpha(x)\textit{-beh-inv}) \\
& \iff (\mathfrak{M}_\circ + \mathfrak{N}_\circ)^\circ \models \alpha[\iota(s)] & \textit{(elem)} \\
& \iff \mathfrak{M} \models \alpha[s] & \textit{(bm+st)}
\end{array}$$ QED

5.5.7. Remark. Note that in the proof of Theorem 5.5.5, we could have assumed $\alpha(x)$ to be invariant for any of the three equivalence notions, since Proposition 5.3.18 tells us that also $s_U \sim t_U$ and $s_U \sim_p t_U$. ◁

5.5.8. Remark. An analogue of Van Benthem's theorem for monotonic modal logic was proved by Pauly (see [113, 52]). Although the translation of monotonic modal logic and monotonic neighbourhood models is very similar to ours, Pauly's approach is slightly different to the present one, since his result is not formulated relative to the class of first-order models which are the translation monotonic models. Rather, he defines a notion of monotonic bisimulation which applies to all first-order $\mathcal{L}_1$-models, and shows that translations of monotonic modal formulas are invariant under this bisimulation notion, even if the first-order models involved are not necessarily translations of monotonic models. This means his result concerns a stronger notion of invariance. The converse is shown using ω-saturation and monotonic modal saturation, and is similar to the proof of the Van Benthem theorem. We do not get a characterisation theorem for monotonic modal logic (relative to translations of monotonic models) as a direct corollary of Theorem 5.5.5, but we believe it is possible to prove one using the same line of argumentation and constructions. ◁

5.5.9. Remark. It seems straightforward to generalise Theorem 5.5.5 to multi-modal classical modal logic with polyadic modalities of finite arity. Multi-modal neighbourhood models are of interest in coalgebraic modal logic due to the following:

It is not always possible to find a collection of separating unary, predicate liftings for a functor $F\colon \mathsf{Set} \to \mathsf{Set}$. However, Schröder showed in [136] that any finitary functor F has a separating set of finitary, *polyadic* predicate liftings, i.e., there exists a finitary coalgebraic modal logic with polyadic modalities which is expressive for F-coalgebras. A k-ary predicate lifting $\lambda\colon (2^{(-)})^k \to 2^{F(-)}$ has transposite $\hat{\lambda}_X\colon F(X) \to N^k(X)$, where we denote by N^k the functor defined by $N^k = 2^{(-)} \circ (2^{(-)})^k$. Note that a map $X \to N^k(X)$ is a k-ary neighbourhood function.

If Λ is a separating set of k-ary predicate liftings for F, then for all sets X, the source of transposites $\{\hat{\lambda}_X\colon F(X) \to N^k(X) \mid \lambda \in \Lambda\}$ yields a natural embedding.

$$\langle \hat{\lambda} \rangle_{\lambda \in \Lambda}\colon F \to \Pi_\Lambda N^k, \tag{5.13}$$

where $\Pi_\Lambda N^k$ is the $|\Lambda|$-fold product of N^k. Hence for every finitary functor F, an F-coalgebra can be transformed into a pointwise equivalent multi-modal, polyadic neighbourhood frame. $\lhd$

5.5.3 Interpolation

In this section we show that the results on ultrafilter extensions from the previous section can be used to prove Craig interpolation for classical modal logic. For several normal and monotonic modal logics, Craig interpolation can be proved using superamalgamation in the corresponding variety of modal algebras, see e.g. [43, 56, 92, 93, 94]. We believe similar proofs can be carried out for classical modal logic. Our proof, however, is based on the ideas used in the proof of Craig interpolation for normal modal logic presented in [8]. The proof in [8] uses first-order model-theoretic arguments similar to those employed in the proof of the Van Benthem characterisation theorem, but Theorem 5.4.26 allows us to prove Craig interpolation in a purely modal setting, without the use of ω-saturated models or the explicit use of algebraic duality. All that is needed is that modal truth is invariant under ultrafilter extensions (Lemma 5.4.23), and that ultrafilter extensions are modally saturated (Proposition 5.4.24).

So far we have worked with a fixed a set At of atomic propositions, giving rise to the language $\mathcal{L} = \mathcal{L}(\mathsf{At})$. In the current section we need to generalise our notions of bounded morphism and modal saturation to sublanguages $\mathcal{L}(\mathsf{At}')$ of $\mathcal{L}(\mathsf{At})$ generated by a specific subset At' of atomic propositions. We point out that all models are always models for the full language $\mathcal{L}(\mathsf{At})$. This generalisation is straightforward, but in the interest of clarity we provide the details and the exact results we need. Let $\mathsf{At}' \subseteq \mathsf{At}$, and let $\mathcal{M}_1 = \langle S_1, \nu_1, V_1\rangle$ and $\mathcal{M}_2 = \langle S_2, \nu_2, V_2\rangle$ be neighbourhood $\mathcal{L}(\mathsf{At})$-models. A function $f\colon S_1 \to S_2$ is a *bounded $\mathcal{L}(\mathsf{At}')$-morphism from $\mathcal{M}_1$ to $\mathcal{M}_2$* (notation: $f\colon \mathcal{M}_1 \to_{\mathcal{L}(\mathsf{At}')} \mathcal{M}_2$) if f is a bounded (frame) morphism from $\langle S_1, \nu_1\rangle$ to $\langle S_2, \nu_2\rangle$, and for all $p \in \mathsf{At}'$, and all $s \in S_1$: $s \in V_1(p)$ iff $f(s) \in V_2(p)$. An $\mathcal{L}(\mathsf{At}')$-congruence is the kernel of a bounded $\mathcal{L}(\mathsf{At}')$-morphism. Two states $s_1 \in S_1$ and $s_2 \in S_2$ are *modally $\mathcal{L}(\mathsf{At}')$-equivalent* (notation: $s_1 \equiv_{\mathcal{L}(\mathsf{At}')} s_2$), if they satisfy the same $\mathcal{L}(\mathsf{At}')$-formulas. Given a neighbourhood $\mathcal{L}(\mathsf{At})$-model $\mathcal{M} = \langle S, \nu, V\rangle$, a subset $X \subseteq S$ is *modally $\mathcal{L}(\mathsf{At}')$-compact* if for all sets Ψ of modal $\mathcal{L}(\mathsf{At}')$-formulas, Ψ is satisfiable in X, whenever Ψ is finitely satisfiable in X, and $\mathcal{M}$ is *modally $\mathcal{L}(\mathsf{At}')$-saturated* if for every $\equiv_{\mathcal{L}(\mathsf{At}')}$-coherent neighbourhood X, both X and X^c are modally $\mathcal{L}(\mathsf{At}')$-compact.

5.5.10. Lemma. *Let $\mathsf{At}' \subseteq \mathsf{At}$.*

1. *If $\mathcal{M}_1$ and $\mathcal{M}_2$ are $\mathcal{L}(\mathsf{At})$-neighbourhood models, and $f\colon \mathcal{M}_1 \to_{\mathcal{L}(\mathsf{At}')} \mathcal{M}_2$, then for all s in $\mathcal{M}_1$, and all $\varphi \in \mathcal{L}(\mathsf{At}')$: $\mathcal{M}_1, s \models \varphi$ iff $\mathcal{M}_2, f(s) \models \varphi$.*

2. *If* $\mathcal{M} = \langle S, \nu, V\rangle$ *is a neighbourhood* $\mathcal{L}(\mathsf{At})$*-model, and* $R \subseteq S \times S$ *is an equivalence relation, then* R *is an* $\mathcal{L}(\mathsf{At}')$*-congruence on* $\mathcal{M}$ *iff* R *is a congruence on the underlying frame* $\langle S, \nu\rangle$*, and for all* $\langle s, t\rangle \in R$*, and all* $p \in \mathsf{At}'$*:* $s \in V(p)$ *iff* $t \in V(p)$*.*

3. *If a neighbourhood* $\mathcal{L}(\mathsf{At})$*-model* $\mathcal{M}$ *is modally* $\mathcal{L}(\mathsf{At}')$*-saturated, then all* $\equiv_{\mathcal{L}(\mathsf{At}')}$*-coherent subsets are definable by an* $\mathcal{L}(\mathsf{At}')$*-formula.*

4. *If a neighbourhood* $\mathcal{L}(\mathsf{At})$*-model* $\mathcal{M}$ *is modally* $\mathcal{L}(\mathsf{At}')$*-saturated, then the relation* $\equiv_{\mathcal{L}(\mathsf{At}')}$ *is an* $\mathcal{L}(\mathsf{At}')$*-congruence.*

5. *If* $\mathcal{M}$ *is neighbourhood* $\mathcal{L}(\mathsf{At})$*-model, then its ultrafilter extension* $\mathcal{M}^u$ *is modally* $\mathcal{L}(\mathsf{At}')$*-saturated.*

Proof. As usual, 1 can be proved by straightforward formula induction. Item 2 is immediate. Item 3 can be proved by retracing the argument used in Lemma 5.4.5. Item 4 follows from item 3 and essentially the same argument used in Lemma 5.4.3. Item 5 can be proved in the same way as Proposition 5.4.24. QED

For a formula $\varphi \in \mathcal{L}$, we denote by $\mathsf{At}(\varphi)$ the set of atomic propositions occurring in φ. Recall that for $\Phi \cup \{\varphi\} \subseteq \mathcal{L}$, we write $\Phi \models \varphi$ if φ is a local semantic consequence of Φ over the class of all neighbourhood models. Note that compactness of $\models$ follows from the compactness of $\models_{\mathbf{N}}$, the first-order consequence relation over the class of neighbourhood models.

5.5.11. Theorem (Interpolation). *Let* $\varphi_1, \varphi_2 \in \mathcal{L}$*. If* $\models \varphi_1 \to \varphi_2$*, then there exists a formula* $\chi \in \mathcal{L}$ *with* $\mathsf{At}(\chi) \subseteq \mathsf{At}(\varphi_1) \cap \mathsf{At}(\varphi_2)$ *such that* $\models \varphi_1 \to \chi$ *and* $\models \chi \to \varphi_2$*.*

Proof. Assume that $\models \varphi_1 \to \varphi_2$. Let $\mathsf{At}_i = \mathsf{At}(\varphi_i)$, $i = 1, 2$, and $\mathsf{At}_0 = \mathsf{At}_1 \cap \mathsf{At}_2$. Denote by $\mathrm{Cons}_{\mathcal{L}(\mathsf{At}_0)}(\varphi_1) = \{\chi \in \mathcal{L}(\mathsf{At}_0) \mid \varphi_1 \models \chi\}$ the set of modal $\mathcal{L}(\mathsf{At}_0)$-consequences of φ_1. It suffices to show that $\mathrm{Cons}_{\mathcal{L}(\mathsf{At}_0)}(\varphi_1) \models \varphi_2$, since then by compactness, there are $\chi_1, \ldots, \chi_n \in \mathrm{Cons}_{\mathcal{L}(\mathsf{At}_0)}(\varphi_1)$ such that $\chi_1 \wedge \ldots \wedge \chi_n \models \varphi_2$, and $\varphi_1 \models \chi_1 \wedge \ldots \wedge \chi_n$, i.e, $\chi = \chi_1 \wedge \ldots \wedge \chi_n$ is a Craig interpolant for $\varphi_1 \to \varphi_2$.

So, assume $\mathcal{M}$ is an $\mathcal{L}(\mathsf{At})$-model and s is a state in $\mathcal{M}$ such that $\mathcal{M}, s \models \mathrm{Cons}_{\mathcal{L}(\mathsf{At}_0)}(\varphi_1)$, and let $\Psi = \{\psi \in \mathcal{L}(\mathsf{At}_0) \mid \mathcal{M}, s \models \psi\}$. Now $\Psi \cup \{\varphi_1\}$ is consistent, since otherwise there would exist $\{\psi_1, \ldots, \psi_n\} \subseteq \Psi$ such that $\models \psi_1 \wedge \ldots \wedge \psi_n \to \neg\varphi_1$, hence $\models \varphi_1 \to \neg\psi_1 \vee \ldots \vee \neg\psi_n$, which would imply that $\neg\psi_1 \vee \ldots \vee \neg\psi_n \in \mathrm{Cons}_{\mathcal{L}(\mathsf{At}_0)}(\varphi_1)$ contradicting the assumption that $\mathcal{M}, s \models \mathrm{Cons}_{\mathcal{L}(\mathsf{At}_0)}(\varphi_1)$.

By definition of $\models$, $\Psi \cup \{\varphi_1\}$ is satisfiable in some neighbourhood $\mathcal{L}(\mathsf{At})$-model $\mathcal{N}$ at a state t in $\mathcal{N}$, i.e., $\mathcal{N}, t \models \Psi \cup \{\varphi_1\}$. Then by construction $s \equiv_{\mathcal{L}(\mathsf{At}_0)} t$, and

as truth is preserved by the injections $\iota\colon \mathcal{M} \to \mathcal{N} + \mathcal{M}$ and $\kappa\colon \mathcal{N} \to \mathcal{N} + \mathcal{M}$, and when passing to ultrafilter extensions, the principal ultrafilters generated by $\iota(s)$ and $\kappa(t)$ are also modally $\mathcal{L}(\mathsf{At}_0)$-equivalent in $\mathcal{U} = \langle U, \mu, V\rangle = (\mathcal{N} + \mathcal{M})^u$, i.e., $\mathsf{u}_{\iota(s)} \equiv_{\mathcal{L}(\mathsf{At}_0)} \mathsf{u}_{\kappa(t)}$. Now since ultrafilter extensions are modally $\mathcal{L}(\mathsf{At}_0)$-saturated (Lemma 5.5.10(5)) it follows from Lemma 5.5.10(4) that $\equiv_{\mathcal{L}(\mathsf{At}_0)}$ is an $\mathcal{L}(\mathsf{At}_0)$-congruence on $\mathcal{U}$. For ease of notation, we denote the relation $\equiv_{\mathcal{L}(\mathsf{At}_0)}$ on $\mathcal{U}$ by Z in the rest of this proof. We have, in particular, Z is a congruence on the underlying frame $\langle U, \mu\rangle$ of $\mathcal{U}$, and by Proposition 5.3.18 Z is also a $\mathcal{2}^{\mathcal{2}}$-bisimulation on $\langle U, \mu\rangle$. This means there exists a coalgebra map $\zeta\colon Z \to \mathcal{2}^{\mathcal{2}}(Z)$ such that the projections $\pi_i\colon \langle Z, \zeta\rangle \to \langle U, \mu\rangle$, $i = 1, 2$, are bounded frame morphisms. We now define a valuation V' on $\langle Z, \zeta\rangle$ to obtain a neighbourhood $\mathcal{L}(\mathsf{At})$-model $\mathcal{Z} = \langle Z, \zeta, V'\rangle$ such that $\pi_1\colon \mathcal{Z} \to \mathcal{U}$ is a bounded $\mathcal{L}(\mathsf{At}_1)$-morphism and $\pi_2\colon \mathcal{Z} \to \mathcal{U}$ is a bounded $\mathcal{L}(\mathsf{At}_2)$-morphism. Let $p \in \mathsf{At}$ and $\langle u_1, u_2\rangle \in Z$, then we define

$$\langle u_1, u_2\rangle \in V'(p) \iff \begin{cases} u_1 \in V(p) & \text{if } p \in \mathsf{At}_1, \\ u_2 \in V(p) & \text{if } p \in \mathsf{At}_2, \\ \text{never} & \text{if } p \in \mathsf{At} \setminus (\mathsf{At}_1 \cup \mathsf{At}_2). \end{cases}$$

Note that V' is well-defined due to Lemma 5.5.10(2). The construction is illustrated below. The dashed arrow going to $\mathcal{U}$ indicates that the principal ultrafilter map u is not a bounded morphism, still u does preserve modal truth (Lemma 5.4.23).

$$\varphi_1 \mathrel{=\!|} \mathcal{N}, t \xrightarrow{\ \kappa\ } \mathcal{N} + \mathcal{M} \xleftarrow{\ \iota\ } \mathcal{M}, s \models \mathrm{Cons}_{\mathcal{L}(\mathsf{At}_0)}(\varphi_1)$$

$$\begin{array}{ccc} & & \mathcal{N} + \mathcal{M} \\ & & \mathsf{u} \mid \equiv \\ & & \mathcal{U} \\ & \pi_1 \swarrow \quad \searrow \pi_2 & \\ & \mathcal{Z} & \end{array}$$

Now we have: $\mathcal{N}, t \models \varphi_1$ implies $\mathcal{U}, \mathsf{u}_{\kappa(t)} \models \varphi_1$. Since $\langle \mathsf{u}_{\kappa(t)}, \mathsf{u}_{\iota(s)}\rangle \in Z$ and π_1 is a bounded $\mathcal{L}(\mathsf{At}_1)$-morphism from $\mathcal{Z}$ to $\mathcal{U}$, we have $\mathcal{Z}, \langle \mathsf{u}_{\kappa(t)}, \mathsf{u}_{\iota(s)}\rangle \models \varphi_1$. By the main assumption that $\models \varphi_1 \to \varphi_2$, we get that $\mathcal{Z}, \langle \mathsf{u}_{\kappa(t)}, \mathsf{u}_{\iota(s)}\rangle \models \varphi_2$, and now since π_2 is a bounded $\mathcal{L}(\mathsf{At}_2)$-morphism from $\mathcal{Z}$ to $\mathcal{U}$, we get $\mathcal{U}, \mathsf{u}_{\iota(s)} \models \varphi_2$ and hence $\mathcal{M}, s \models \varphi_2$. QED

5.6 Conclusion and related work

In the first part of this chapter we discussed and compared different notions of equivalence between neighbourhood structures. We gave back-and-forth style characterisations of $\mathcal{2}^{\mathcal{2}}$-bisimulations and precocongruences, and showed that,

as expected, behavioural equivalence is the only one of the three notions that allows us to prove a Hennessy-Milner theorem for image-finite neighborhood models (cf. Section 5.4). Furthermore, we showed that for an *arbitrary* Set-functor F, precocongruences capture behavioural equivalence on a single F-coalgebra (Theorem 5.3.11). For functors F that weakly preserve kernel pairs, such as $\mathcal{2}^{\mathcal{2}}$, this is already achieved with F-bisimulations [51], but we believe that precocongruences could be an interesting alternative to F-bisimulations for functors which lack this property. A first indication of this is [81] where precocongruences are used to obtain a game-theoretic characterisation of behavioural equivalence.

After having reached a good understanding of state equivalence over neighbourhood structures, we focused on generalising two well-known model-theoretic results in normal modal logic to classical modal logic: the Van Benthem characterisation theorem (Theorem 5.5.5) and the Craig interpolation theorem (Theorem 5.5.11). Our proof of Theorem 5.5.5 builds on ideas from the original proof of the Van Benthem characterisation theorem ([20]). Closely related to our work are also the invariance results by Pauly ([113]) on monotonic modal logic, and Ten Cate et al. ([32]) on topological modal logic.

A number of other model-theoretic results are worth exploring. Perhaps the most interesting one is a generalisation of the Goldblatt-Thomason Theorem (see e.g. [25]). The classic result for Kripke models can be proved using model-theoretic constructs or by using algebraic duality. The algebraic duality proof has already been generalised to the coalgebraic setting by Kurz & Rosický's [89]. Indeed, a special case of their main result is the result we are after: a Goldblatt-Thomason Theorem for neighbourhood models (cf. [89], Corollary 3.17(2) and Remark 3.18). Given the formal machinery we have developed in this paper (e.g., the ultrafilter extensions from Section 5.4.3), one may hope for a model-theoretic proof of this result (see e.g., Section 3.8 in [25]). Such a model-theoretic proof has been given for topological models (which are special cases of neighbourhood models) by Ten Cate et al. ([32]). However, an important ingredient in the model-theoretic proof for the Kripke case is the fact that any Kripke model is bisimilar to the disjoint union of its generated submodels. This is not true for an arbitrary neighbourhood model (cf. [49]), and at the moment, it is not clear which alternative construction could be used in its place.

A second model-theoretic issue raised by the results in this paper concerns our translation of the modal language into a two-sorted first-order language (cf. Definition 5.5.2). As is well-known, with respect to Kripke structures, the basic modal language can be translated into the *guarded fragment* of first-order logic (cf. [8]). This fact has been used to explain a number of the important properties of modal logic (see, for example, [7] for an extensive discussion). The question is whether classical modal logic is also contained in some kind of guarded fragment. Our translation of $\Box\varphi$ does not fall into the guarded fragment of two-sorted first-order logic. However, it is not difficult to see that over the

class **N** of neighbourhood models viewed as first-order structures, $st_x(\Box\varphi)$ is equivalent to the following single-sorted first-order formula:

$$\exists u(Nbhd(u) \wedge x\mathsf{N}u \wedge \forall y(u\mathsf{E}y \rightarrow st_y(\varphi)) \wedge \forall y(State(y) \rightarrow (\neg(st_y(\varphi)) \vee u\mathsf{E}y)))$$

where $Nbhd$ and $State$ are designated predicates intended to mean "...is a neighbourhood" and "...is a state", respectively. This formula is in the (loosely) guarded fragment.

Our characterisation theorem for classical modal logic leads to a number of interesting research questions. For example, we would like to explore the possibility of proving our result using game-theoretic techniques similar to the ones exploited by Otto ([107]). Furthermore, neighbourhood structures can also be seen as a type of Chu spaces. We would like to relate our characterisation theorem to Van Benthem's characterisation of the Chu transform invariant fragment of a two-sorted first-order logic in [21].

Finally, it would be interesting to find out if our characterisation theorem can be generalised to coalgebraic modal logic for an arbitrary finitary functor $F\colon \mathsf{Set} \rightarrow \mathsf{Set}$, using the embedding of F-coalgebras into multi-modal, k-ary neighbourhood frames as described in Remark 5.5.9. It might be possible to prove that, under certain assumptions, the coalgebraic modal logic over F-coalgebras can be viewed as the bisimulation invariant fragment of some many-sorted first-order logic. Initial investigations suggest that this is possible for functors of the form $A^{(2^k)^{(-)}}$ where A is a finite set and k is a natural number. An $A^{(2^k)^{(-)}}$-coalgebra can be seen as a multi-modal, polyadic neighbourhood frame $\langle X, \{\nu_a \mid a \in A\}\rangle$ given by an A-indexed collection of k-ary neighbourhood functions $\nu_a\colon X \rightarrow 2^{(2^X)^k}$ such that for each k-tuple of subsets $\langle U_1, \ldots, U_k\rangle$ and each state $x \in X$, $\langle U_1, \ldots, U_k\rangle \in \nu_a(x)$ for exactly one $a \in A$. We must leave the details of this result as future work.

Bibliography

[1] P. Aczel and N.P. Mendler. A final coalgebra theorem. In D.H. Pitt, D.E. Rydeheard, P. Dybjer, A.M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365, 1989.

[2] J. Adámek. *Theory of Mathematical Structures*. D. Reidel Publishing Company, 1983.

[3] J. Adámek, H. Herrlich, and G.E. Strecker. *Abstract and Concrete Categories: The Joy of Cats*. J. Wiley and Sons, 1990. Online version: http://katmat.math.uni-bremen.de/acc.

[4] J. Adámek and H.-E. Porst. On tree coalgebras and coalgebra presentations. *Theoretical Computer Science*, 311:257–283, 2004.

[5] J. Adámek and V. Trnková. *Automata and Algebras in Categories*. Kluwer Academic Publishers, 1990.

[6] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[7] H. Andréka, J. van Benthem, and I. Németi. Back and forth between modal logic and classical logic. *Logic Journal of the IGPL*, 3:685–720, 1995.

[8] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[9] V. Antimirov. Partial derivatives of regular expressions, and finite automaton constructions. *Theoretical Computer Science*, 155(2):291–319, 1996.

[10] F. Arbab and J.J.M.M. Rutten. A coinductive calculus of component connectors. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *Recent Trends in Algebraic Development Techniques. Proceedings of WADT 2002*, volume 2755 of *Lecture Notes in Computer Science*, pages 35–56. Springer, 2003.

[11] A. Aziz, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. Sequential synthesis using S1S. *IEEE Transactions on Computer-Aided Design of Integrated Citcuits and Systems*, 19(10):1149–1162, 2000.

[12] C. Baier and J.-P. Katoen. *Principles of Model-Checking*. The MIT Press, 2008.

[13] L.S. Barbosa. *Components as Coalgebras*. PhD thesis, Universidade do Minho, 2001.

[14] M. Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114:299–315, 1993.

[15] F. Bartels, A. Sokolova, and E. de Vink. A hierarchy of probabilistic system types. In H.P. Gumm, editor, *Proceedings of the 6th Workshop on Coalgebraic Methods in Computer Science (CMCS 2003)*, volume 82(1) of *Electronic Notes in Theoretical Computer Science*, pages 57–75. Elsevier Science Publishers, 2003.

[16] F. Bartels, A. Sokolova, and E. de Vink. A hierarchy of probabilistic system types. *Theoretical Computer Science*, 327:3–22, 2004.

[17] M.-P. Béal and O. Carton. Computing the prefix of an automaton. *Theoretical Informatics and Applications (RAIRO)*, 34(6):503–514, 2000.

[18] M.-P. Béal and O. Carton. Determinization of transducers over finite and infinite words. *Theoretical Computer Science*, 289:225–251, 2002.

[19] J. van Benthem. *Modal Correspondence Theory*. PhD thesis, Universiteit van Amsterdam, 1976.

[20] J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Extensions of Classical Logic*, volume II of *Handbook of Philosophical Logic*, pages 167–247. Reidel, 1984.

[21] J. van Benthem. Information transfer across Chu spaces. *Logic Journal of the IGPL*, 8(6):719–731, 2000.

[22] G. Berry and R. Sethi. From regular expressions to deterministic automata. *Theoretical Computer Science*, 48:117–126, 1986.

[23] J. Berstel. *Transductions and Context-Free Languages*, volume 38 of *Leitfäden der angewandten Mathematik und Mechanik*. B. G. Teubner, Stuttgart, 1979.

[24] P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, 2007.

[25] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[26] M.M. Bonsangue and A. Kurz. Presenting functors by operations and equations. In L. Aceto and A. Ingolfsdottir, editors, *Proceedings of Foundations of Software Science and Computations Structures (FoSSaCS 2006)*, volume 3921 of *Lecture Notes in Computer Science*, pages 172–186, 2006.

[27] M.M. Bonsangue, J.J.M.M. Rutten, and A.M. Silva. Coalgebraic logic and synthesis of Mealy machines. In *Proceedings of Foundations of Software Science and Computations Structures (FoSSaCS 2008)*, volume 4962 of *Lecture Notes in Computer Science*, pages 231–245. Springer, 2008.

[28] M.M. Bonsangue, J.J.M.M. Rutten, and A.M. Silva. Algebras for Kripke polynomial coalgebras. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS 2009)*, 2009. To appear.

[29] V. Bruyère and C. Reutenauer. A proof of Choffrut's theorem on subsequential functions. *Theoretical Computer Science*, 215:329–335, 1999.

[30] J.A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.

[31] J.R. Büchi and L.H.G. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[32] B. ten Cate, D. Gabelaia, and D. Sustretov. Modal languages for topology: Expressivity and definability. *Annals of Pure and Applied Logic*. To appear.

[33] J.-M. Champarnaud and D. Ziadi. Canonical derivatives, partial derivatives and finite automaton constructions. *Theoretical Computer Science*, 289:137–163, 2001.

[34] C. Chang and H. Keisler. *Model Theory*. North-Holland, 1973.

[35] B.F. Chellas. *Modal Logic—An Introduction*. Cambridge University Press, 1980.

[36] C. Choffrut. A generalization of Ginsburg and Rose's characterization of g-s-m mappings. In H.A. Maurer, editor, *Prooceedings of the 6th International Colloquium on Automata, Languages and Programming (ICALP 1979)*, volume 71 of *Lecture Notes in Computer Science*, pages 88–103, 1979.

[37] C. Choffrut. Minimizing subsequential transducers: A survey. *Theoretical Computer Science*, 292:131–143, 2003.

[38] A. Church. Logic, arithmetics and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1963.

[39] C. Cîrstea and D. Pattinson. Modular construction of complete coalgebraic logics. *Theoretical Computer Science*, 388:83–108, 2007.

[40] K. Došen. Duality between modal algebras and neighbourhood frames. *Studia Logica*, 48:219–234, 1989.

[41] S. Eilenberg. *Automata, Languages and Machines (Vol. A)*. Academic Press, 1974.

[42] J. Flum and M. Ziegler. *Topological Model Theory*, volume 769 of *Lecture Notes in Mathematics*. Springer, 1980.

[43] D.M. Gabbay and L. Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logic*. Number 46 in Oxford Logic Guides. Oxford University Press, 2005.

[44] F.D. Garcia, I. Hasuo, W. Pieters, and P. van Rossum. Provable anonymity. In R. Küsters and J. Mitchell, editors, *Proceedings of the 3rd ACM Workshop on Formal Methods in Security Engineering (FMSE 2005)*, pages 63–72. ACM Press, 2005.

[45] M. Gehrke, S. Grigorieff, and J.-É Pin. Duality and equational theory of regular languages. In L. Aceto et al., editors, *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*, volume 5126 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2008.

[46] L. Goble. Murder most gentle: The paradox deepens. *Philosophical Studies*, 64(2):217–227, 1991.

[47] F.Q. Gouvêa. *p-adic Numbers: An Introduction*. Springer, 1993.

[48] D. Gries. Describing an algorithm by Hopcroft. *Acta Informatica*, 2:97–109, 1973.

[49] H.P. Gumm. Functors for coalgebras. *Algebra Universalis*, 45:135–147, 2001.

[50] H.P. Gumm. On minimal coalgebras. *Applied Categorical Structures*, 2008. To appear.

[51] H.P. Gumm and T. Schröder. Types and coalgebraic structure. *Algebra Universalis*, 53:229–252, 2005.

[52] H.H. Hansen. Monotonic modal logic (Master's thesis). Research Report PP-2003-24, Institute for Logic, Language and Computation. Universiteit van Amsterdam, 2003.

[53] H.H. Hansen. Coalgebraising subsequential transducers. In *Proceedings of the 9th Workshop on Coalgebraic Methods in Computer Science (CMCS 2008)*, volume 203(5) of *Electronic Notes in Theoretical Computer Science*, pages 109–129. Elsevier Science Publishers, 2008.

[54] H.H. Hansen and D. Costa. DIFFCAL. Tool webpage (source code, documentation, executable) currently available at: `http://homepages.cwi.nl/~costa/projects/diffcal`, 2005. Implementation of Mealy synthesis algorithm described in [55].

[55] H.H. Hansen, D. Costa, and J.J.M.M. Rutten. Synthesis of Mealy machines using derivatives. In *Proceedings of the 8th Workshop on Coalgebraic Methods in Computer Science (CMCS 2006)*, volume 164(1) of *Electronic Notes in Theoretical Computer Science*, pages 27–45. Elsevier Science Publishers, 2006.

[56] H.H. Hansen and C. Kupke. A coalgebraic perspective on monotone modal logic. In *Proceedings of the 7th Workshop on Coalgebraic Methods in Computer Science (CMCS 2004)*, volume 106 of *Electronic Notes in Theoretical Computer Science*, pages 121–143. Elsevier Science Publishers, 2004.

[57] H.H. Hansen., C. Kupke, and E. Pacuit. Bisimulation for neighbourhood structures. In T. Mossakowski, U. Montanari, and M. Haveraaen, editors, *Proceedings of the 2nd Conference on Algebra and Coalgebra in Computer Science (CALCO 2007)*, volume 4624 of *Lecture Notes in Computer Science*, pages 279–293. Springer, 2007.

[58] H.H. Hansen., C. Kupke, and E. Pacuit. Neighbourhood structures: bisimilarity and basic model theory. *Logical Methods in Computer Science*, 2009. To appear.

[59] D. Harel and A. Pnueli. On the development of reactive systems. In K. Apt, editor, *Logics and Models of Concurrent Systems*, volume F-13 of *NATO-ASI Series*, pages 477–498. Springer, New York, 1985.

[60] I. Hasuo and B. Jacobs. Context-free languages via coalgebraic trace semantics. In J.L. Fiadero, N. Harman, M. Roggenbach, and J.J.M.M. Rutten, editors, *Proceedings of the 1st Conference on Algebraic and Coalgebraic Methods in Computer Science (CALCO 2005)*, volume 3629 of *Lecture Notes in Computer Science*, pages 175–193. Springer, 2005.

[61] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4:11), 2007.

[62] I. Hasuo, B. Jacobs, and A. Sokolova. The microcosm principle and concurrency in coalgebra. In *Proceedings of Foundations of Software Science and Computation Structures (FoSSaCS 2008)*, volume 4962 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2008.

[63] I. Hasuo and Y. Kawabe. Probabilistic anonymity via coalgebraic simulations. In *Proceedings of the European Symposium on Programming (ESOP 2007)*, volume 4421 of *Lecture Notes in Computer Science*, pages 379–394. Springer, 2007.

[64] T.A. Henzinger, S.C. Krishnan, O. Kupferman, and F.Y.C. Mang. Synthesis of uninitialized systems. In *Proceedings of the 29th International Colloqium on Automata, Languages and Programming (ICALP 2002)*, volume 2380 of *Lecture Notes in Computer Science*, pages 644–656. Springer, 2002.

[65] J.E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi and Z. Paz, editors, *Theory of Machines and Computation*, pages 189–196. Academic Press, 1971.

[66] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 2nd edition, 2001.

[67] B. Jacobs. Objects and classes, co-algebraically. In B. Freitag, C.B. Jones, C. Lengauer, and H.-J. Schek, editors, *Object-Orientation with Parallelism and Persistence*, pages 83–103. Kluwer Academic Publishers, 1996.

[68] B. Jacobs. Many-sorted coalgebraic modal logic: a model-theoretic study. *Theoretical Informatics and Applications (RAIRO)*, 35:31–59, 2001.

[69] B. Jacobs. *Introduction to Coalgebra. Towards Mathematics of States and Observations*. Book draft, 2005.

[70] B. Jacobs. A bialgebraic review of deterministic automata, regular expressions and languages. In K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Algebra, Meaning and Computation: Essays dedicated to Joseph A. Goguen on the Occasion of his 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 375–404. Springer, 2006.

[71] D. Janin and I. Walukiewicz. On the expressive completeness of the modal mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *Proceedings of 7th International Conference on Concurrency Theory (CONCUR 1996)*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1996.

[72] R.H. Katz. *Contemporary Logic Design*. Prentice-Hall, 2nd edition, 2004.

[73] D.E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, 3rd edition, 1997.

[74] T. Knuutila. Re-describing an algorithm by Hopcroft. *Theoretical Computer Science*, 250:333–363, 2001.

[75] N. Koblitz. *p-adic Numbers, p-adic Analysis, and Zeta-functions*, volume 58 of *Graduate Texts in Mathematics*. Springer, 1977.

[76] Z. Kohavi. *Switching and Finite Automata Theory*. Computer Science Series. McGraw-Hill Higher Education, 2nd edition, 1990.

[77] D. Kozen. *Automata and Computability*. Undergraduate Texts in Computer Science. Springer, 1997.

[78] M. Kracht and F. Wolter. Normal monomodal logics can simulate all others. *Journal of Symbolic Logic*, 64(1):99–138, 1999.

[79] O. Kupferman, N. Piterman, and M.Y. Vardi. Safraless compositional synthesis. In *Proceedings of Computer Aided Verification (CAV 2006)*, volume 4144 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2006.

[80] C. Kupke. *Finitary Coalgebraic Logics*. PhD thesis, Universiteit van Amsterdam, 2006.

[81] C. Kupke. Terminal sequence induction via games. In *Proceedings of the 7th International Tbilisi Symposium on Language, Logic and Computation*, 2008.

[82] C. Kupke, A. Kurz, and D. Pattinson. Algebraic semantics for coalgebraic logics. In *Proceedings of the 7th Workshop on Coalgebraic Methods in Computer Science (CMCS 2004)*, volume 106 of *Electronic Notes in Theoretical Computer Science*, pages 219–241. Elsevier Science Publishers, 2004.

[83] C. Kupke, A. Kurz, and D. Pattinson. Ultrafilter extensions for coalgebras. In J.L. Fiadero, N. Harman, M. Roggenbach, and J.J.M.M. Rutten, editors, *Proceedings of the 1st Conference on Algebraic and Coalgebraic Methods in Computer Science (CALCO 2005)*, volume 3629 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2005.

[84] C. Kupke, A. Kurz, and Y. Venema. Stone coalgebras. *Theoretical Computer Science*, 327((1-2)):109–134, 2004.

[85] C. Kupke, A. Kurz, and Y. Venema. A complete coalgebraic logic. In C. Areces and R. Goldblatt, editors, *Advances in Modal Logic, Volume 7 (AiML 2008)*, pages 193–217. College Publications, 2008.

[86] C. Kupke and Y. Venema. Coalgebraic automata theory: basic results. *Logical Methods in Computer Science*, 4(4:10), 2008.

[87] A. Kurz. *Logics for Coalgebras and Applications to Computer Science*. PhD thesis, Ludwig-Maximilians-Universität, 2000.

[88] A. Kurz. Coalgebras and their logics. *ACM SIGACT News, Logic Column*, 37(2):57–77, 2006.

[89] A. Kurz and J. Rosický. The Goldblatt-Thomason-theorem for coalgebras. In T. Mossakowski, U. Montanari, and M. Haveraaen, editors, *Proceedings of the 2nd Conference on Algebra and Coalgebra in Computer Science (CALCO 2007)*, volume 4624 of *Lecture Notes in Computer Science*, pages 342–355. Springer, 2007.

[90] J. Lambek. A fixpoint theorem for complete categories. *Mathematische Zeitschrift*, 103(2):151–161, 1968.

[91] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer, 1998.

[92] J. Madarász. Interpolation and amalgamation; pushing the limits. Part I. *Studia Logica*, 61:311–345, 1998.

[93] J. Madarász. Interpolation and amalgamation; pushing the limits. Part II. *Studia Logica*, 62:1–19, 1999.

[94] M. Marx. *Algebraic Relativization and Arrow Logic*. PhD thesis, Universiteit van Amsterdam, 1995.

[95] R.F. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, 9(1):39–47, 1960.

[96] G.H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34:1045–1079, September 1955.

[97] R. Milner. *A Calculus of Communicating Processes*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[98] M. Mohri. Finite-state-transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[99] M. Mohri, F.C.N. Pereira, and M. Riley. Speech recognition with weighted finite-state transducers. In L. Rabiner and F. Juang, editors, *Handbook of Speech Processing and Speech Communication, Part E: Speech Recognition*. Springer, 2008.

[100] R. Montague. Universal grammar. *Theoria*, 36:373–398, 1970.

[101] E.F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies*, volume 34 of *Annals of Mathematical Studies*, pages 129–153. Princeton University Press, 1956.

[102] L.S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96:277–317, 1999. Erratum in [103].

[103] L.S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 99:241–259, 1999.

[104] L.S. Moss and I.D. Viglizzio. Harsanyi types spaces and final coalgebras constructed from satisfied theories. In *Proceedings of the 7th Workshop on Coalgebraic Methods in Computer Science (CMCS 2004)*, volume 106 of *Electronic Notes in Theoretical Computer Science*, pages 279–295. Elsevier Science Publishers, 2004.

[105] A.L. Oliveira and J.P.M. Silva. Efficient algorithms for the inference of minimum size DFAs. *Machine Learning*, 44:93–119, 2001.

[106] G. Ott and N.H. Feinstein. Design of sequential machines from their regular expressions. *Journal of the ACM*, 8(4):585–600, 1961.

[107] M. Otto. Bisimulation invariance and finite models. In W. Pohlers Z. Chatzidakis, P. Koepke, editor, *Logic Colloquium '02*, volume 27 of *Lecture Notes in Logic*, pages 276–298. Association for Symbolic Logic, 2006.

[108] V. Padmanabhan, G. Governatori, and K. Su. Knowledge assesment: A modal logic approach. In *Proceedings of the 3rd International Workshop on Knowledge and Reasoning for Answering Questions (KRAQ 2007)*, 2007.

[109] D. Park. Concurrency and automata on infinite sequences. In G. Goos and J. Hartmanis, editors, *Theoretical Computer Science, 5th GI-Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183, 1981.

[110] D. Pattinson. An introduction to the theory of coalgebras. Lecture notes accompanying the course at NASSLLI 2003. Available at http://www.pst.ifi.lmu.de/~pattinso/Publications/nasslli.all.ps.gz.

[111] D. Pattinson. Coalgebraic modal logic: Soundness, completeness and decidability of local consequence. *Theoretical Computer Science*, 309:177–193, 2003.

[112] D. Pattinson. Expressive logics for coalgebras via terminal sequence induction. *Notre Dame Journal of Formal Logic*, 45:19–33, 2004.

[113] M. Pauly. Bisimulation for general non-normal modal logic. Manuscript (unpublished), 1999.

[114] M. Pauly. *Logic for Social Software*. PhD thesis, Universiteit van Amsterdam, 2001.

[115] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[116] S. Peyton-Jones, editor. *Haskell 98 Language and Libraries: the Revised Report*. Cambridge University Press, 2003.

[117] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on the Principles of Programming Languages (POPL 1989)*, pages 179–190. ACM Press, 1989.

[118] G.N. Raney. Sequential functions. *Journal of the ACM*, 5(2):177–180, April 1958.

[119] G.N. Raney. Functional composition patterns and power series reversion. *Transactions of the American Mathematical Society*, 94:441–451, 1960.

[120] G.N. Raney. On continued fractions and finite automata. *Mathematische Annalen*, 206:265–283, 1973.

[121] R. Redziejowski. Construction of a deterministic ω-automaton using derivatives. *Theoretical Informatics and Applications (RAIRO)*, 33(2):133–158, 1999.

[122] C. Reutenauer. Subsequential functions: Characterizations, minimization, examples. In *Aspects and Prospects of Theoretical Computer Science. Proceedings of the 6th International Meeting of Young Computer Scientists (IMYCS 1990)*, volume 464 of *LNCS*, pages 62–79, 1990.

[123] C. Reutenauer and M.-P. Schützenberger. Minimization of rational word functions. *SIAM Journal of Computing*, 20(4):669–685, 1991.

[124] M. Rößiger. Coalgebra and modal logic. In H. Reichel, editor, *Proceedings of the 3rd Workshop on Coalgebraic Methods in Computer Science (CMCS 2000)*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 299–320. Elsevier Science Publishers, 2000.

[125] J. Rothe and D. Mašulović. Towards weak bisimulation for coalgebras. In A. Kurz, editor, *Categorical Methods for Concurrency, Interaction, and Mobility*, volume 68(1) of *Electronic Notes in Theoretical Computer Science*, pages 32–46. Elsevier Science Publishers, 2002.

[126] J.J.M.M. Rutten. Automata and coinduction (an exercise in coalgebra). In D. Sangiorgi and R. de Simone, editors, *Proceedings of 9th International Conference on Concurrency Theory (CONCUR 1998)*, volume 1466 of *Lecture Notes in Computer Science*, pages 194–218. Springer, 1998.

[127] J.J.M.M. Rutten. A note on coinduction and weak bisimilarity for While programs. Technical Report SEN-R9826, Centrum voor Wiskunde en Informatica (CWI), 1998.

[128] J.J.M.M. Rutten. Relators and metric bisimulations. In B. Jacobs, L. Moss, H. Reichel, and J.J.M.M. Rutten, editors, *Proceedings of the 1st Workshop on Coalgebraic Methods in Computer Science (CMCS 1998)*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 257–263. Elsevier Science Publishers, 1998.

[129] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249:3–80, 2000.

[130] J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata and power series. *Theoretical Computer Science*, 308(1):1–53, 2003.

[131] J.J.M.M. Rutten. A coinductive calculus of streams. *Mathematical structures in Computer Science*, 15:93–147, 2005.

[132] J.J.M.M. Rutten. Algebraic specification and coalgebraic synthesis of Mealy machines. In *Proceedings of the 2nd International Workshop on Formal Aspects of Component Software (FACS 2005)*, volume 160 of *Electronic Notes in Theoretical Computer Science*, pages 305–319. Elsevier Science Publishers, 2006.

[133] J.J.M.M. Rutten. Coalgebraic foundations of linear systems (an exercise in stream calculus). In T. Mossakowski, U. Montanari, and M. Haveraaen, editors, *Proceedings of the 2nd Conference on Algebraic and Coalgebraic Methods in Computer Science (CALCO 2007)*, volume 4624 of *Lecture Notes in Computer Science*, pages 425–446. Springer, 2007.

[134] S. Safra. On the complexity of ω-automata. *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS 1988)*, pages 319–327, 1988.

[135] C. Sanchez, M. Slanina, H.B. Sipma, and Z. Manna. The Reaction Algebra: A formal language for event correlation. In *Pillars of Computer Science: Essays Dedicated to Boris Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *Lecture Notes in Computer Science*, pages 586–609. Springer, 2008.

[136] L. Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theoretical Computer Science*, 390:230–247, 2008.

[137] L. Schröder and D. Pattinson. PSPACE bounds for rank-1 modal logics. In *Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 231–242, 2006. Extended version to appear in ACM Transactions on Computational Logics.

[138] L. Schröder and D. Pattinson. Rank-1 modal logics are coalgebraic. In W. Thomas and P. Weil, editors, *Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2007)*, volume 4393 of *Lecture Notes in Computer Science*, pages 573–585, 2007. Extended version to appear in Journal of Logic and Computation.

[139] M.P. Schützenberger. Sur un variante des fonctions sequentielles. *Theoretical Computer Science*, 4(1):47–57, February 1977.

[140] D. Scott. Advice on modal logic. In K. Lambert, editor, *Philosophical Problems in Logic*, pages 143–173. Reidel, 1970.

[141] K. Segerberg. *An Esssay in Classical Modal Logic*. Number 13 in Filosofiska Studier. Uppsala Universitet, 1971.

[142] A. Sokolova, E. de Vink, and H. Woracek. Weak bisimulation for action-type coalgebras. In L. Birkedal, editor, *Proceedings of Category Theory and Computer Science (CTCS 2004)*, volume 122 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2005.

[143] M. Sun and L.S. Barbosa. Components as coalgebras: The refinement dimension. *Theoretical Computer Science*, 351:276–294, 2005.

[144] P. Tiňo and J. Šajda. Learning and extracting initial Mealy automata with a modular neural network model. *Neural Computation*, 7(4):822–844, 1995.

[145] M.T. Tu, E. Wolff, and W. Lamersdorf. Genetic algorithms for automated negotiations: A FSM-based application approach. In *Proceedings of the 11th International Workshop on Database and Expert Systems (DEXA 2000)*, page 1029, 2000.

[146] D. Turi and J.J.M.M. Rutten. On the foundation of final semantics: non-standard sets, metric spaces and partial orders. *Mathematical Structures in Computer Science*, 8:481–540, 1998.

[147] T. Uustalu and V. Vene. Primitive (co)recursion and course-of-value (co)iteration. *Informatica (Lithuanian Academy of Science)*, 10(1):5–26, 1999.

[148] M.Y. Vardi. On epistemic logic and logical omniscience. In J. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge (TARK 1986)*, pages 293–305. Morgan Kaufmann, 1986.

[149] M.Y. Vardi. An automata-theoretic approach to fair realizability and synthesis. In P. Wolper, editor, *Proceedings of the 7th International Conference on Computer Aided Verification (CAV 1995)*, volume 939 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 1995.

[150] M.Y. Vardi. Why is modal logic so robustly decidable? In N. Immerman and P. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 149–184. American Mathematical Society, 1997.

[151] M.Y. Vardi. Automata-theoretic techniques for temporal reasoning. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal*

Logic, volume 3 of *Studies in Logic and Practical Reasoning*, pages 971–989. Elsevier, 2006.

[152] Y. Venema. Algebras and coalgebras. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 331–426. Elsevier, 2006.

[153] Y. Venema. Automata and fixed point logic: a coalgebraic perspective. *Information and Computation*, 204:637–678, 2006.

[154] E. de Vink and J.J.M.M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221:271–293, 1999.

[155] J. Vuillemin. On circuits and numbers. *IEEE Transactions on Computers*, 43(8):868–879, 1994.

[156] J. Vuillemin. Finite digital synchronous circuits are characterised by 2-algebraic truth tables. In *Advances in Computing Science—Proceedings of 6th Asian Computing Science Conference (ASIAN 2000)*, volume 1961 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2000.

[157] J. Vuillemin. Digital algebra and circuits. In *Verification: Theory and Practice. Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, volume 2772 of *Lecture Notes in Computer Science*, pages 100–120. Springer, 2003.

[158] R.J. Wieringa. *Design Methods for Reactive Systems*. Morgan Kaufmann, 2002.

[159] H.S. Wilf. *Generatingfunctionology*. Academic Press, 1994. Online version available at: http://www.math.upenn.edu/~wilf/DownldGF.html.

[160] J. Worrell. On the final sequence of a finitary set functor. *Theoretical Computer Science*, 338:184–199, 2005.

Index

Abstract

Coalgebraic Modelling
Applications in Automata Theory and Modal Logic

In this thesis we apply coalgebraic modelling to gain new insights into automata theory and modal logic. Briefly summarised, the main contributions consist of (i) a coalgebraic technique for synthesising Mealy machines from arithmetic bitstream specifications, and results that relate Mealy machines to rational specifications in terms of expressivity and complexity; (ii) a coalgebraic perspective on subsequential transducers, and a systematic classification of reflective subcategories of subsequential structures; (iii) bisimulation notions for neighbourhood structures and model theoretic results for classical modal logic, including interpolation and a characterisation of classical modal logic as the bisimulation invariant fragment of first order logic. These contributions are found in Chapters 3, 4 and 5.

In Chapter 3 we study the relationship between bitstream functions and Mealy machines with binary input and output. Mealy machines and binary arithmetic play an important role in the modelling and specification of sequential circuits. It is well-known that Mealy machines can be modelled as coalgebras and that a final Mealy coalgebra exists. We present a method which given a so-called rational specification of an arithmetic bitstream function constructs a Mealy machine which realises the specification. The idea behind the construction is to define a Mealy structure on the set of specifications, and construct a realisation as the submachine generated by the given specification. Termination and minimality are ensured by working modulo bisimilarity. This method is essentially coalgebraic since the idea, in principle, applies to other coalgebra types and specification languages. In our case, we obtain a Mealy structure on the set of specifications by defining the arithmetic operations coinductively, and we determine bisimilarity of specifications by reduction to normal form. We have implemented this construction method and we give a complexity analysis of the algorithm. Furthermore, we present upper bounds on the number of states

in the constructed Mealy machine, and we show that there exist finite Mealy machines that are not the realisation of a rational specification.

In Chapter 4 we focus on the coalgebraic modelling of subsequential transducers. Subsequential transducers are a type of automata which combines transduction with acceptance, and they are used in the areas of coding theory and language processing. As a class of automata they generalise Mealy machines and classical deterministic automata, both of which have a neat coalgebraic modelling. The underlying structure of a subsequential transducer also looks like a coalgebra, however, it is quite easy to see that the corresponding notion of coalgebra morphism does not fit well with the intended word function semantics. We show that a proper coalgebraic modelling can be obtained for the subclass of normalised subsequential structures. This result provides a new perspective on known results such as the existence of canonical minimal subsequential transducers, and the fact that subsequential transducers can be minimised by first normalising and then identifying bisimilar states. We also show that normalisation, minimisation and taking differentials are reflectors in the category theoretic sense.

In Chapter 5, our starting point is the coalgebraic modelling of neighbourhood structures. Neighbourhood structures are the standard semantic tool used for reasoning about non-normal modal logics. The modal logic of all neighbourhood models is called 'classical modal logic'. Neighbourhood models generalise Kripke models, but they have received far less attention. In particular, so far a notion of bisimulation was lacking. We apply the coalgebraic modelling to define three notions of state equivalence in neighbourhood structures. One of these is a new coalgebraic notion which lies in between bisimilarity and behavioural equivalence. In the modal logic of Kripke models, the close relationship between bisimilarity and the expressivity of the modal language is known from results such as the Hennessy-Milner theorem and Van Benthem's characterisation theorem. We prove analogues of these results for classical modal logic, and we give a model theoretic proof of Craig interpolation in which coalgebraic bisimulations play a central role.

Samenvatting

Coalgebraïsch Modelleren
Toepassingen in de Automatentheorie en de Modale Logica

Het algemene doel van deze dissertatie is om nieuwe inzichten te verwerven in de automatentheorie en de modale logica door de relevante structuren als coalgebras te modelleren. Kort samengevat bestaan de specifieke bijdragen uit een coalgebraïsche techniek voor synthese van Mealy machines uit specificaties van rekenkundige bitstroom functies, het coalgebraïsch modelleren en classificeren van subsequentiële transducers, noties van bisimulatie voor omgevingsstructuren en model-theoretische stellingen voor klassieke modale logica. Deze contributies zijn te vinden in Hoodstuk 3, 4 en 5.

In Hoofdstuk 3 bekijken wij de relatie tussen bitstroom functies en Mealy machines met binaire input en output. Mealy machines en binarie rekenkunde spelen een belangrijke rol in het modelleren en specificeren van sequentiële circuits. Mealy machines zijn bekende voorbeelden van een type automaten die als coalgebras kunnen worden beschouwd, en waarvoor een finale coalgebra bestaat. Wij presenteren een methode die gegeven een zogenoemde rationale specificatie in de binaire rekenkunde, een eindige Mealy machine construeert die de specificatie realiseert. Het idee achter de constructie is om een Mealy-structuur te leggen op de verzameling van specificaties, en een realisatie te construeren als de submachine gegenereerd door de gegeven specificatie. Terminatie en minimaliteit worden gegarandeerd door modulo bisimilariteit te werken. Deze methode is in essentie coalgebraïsch omdat het idee in principe voor andere coalgebra types en specificatietalen ook toepasbaar is. In ons geval wordt de Mealy structuur op de verzameling van specificaties bereikt door de rekenkundige operatoren coïnductief te definiëren, en bisimilariteit wordt beslist door specificaties te reduceren naar normaalvorm. Wij hebben deze constructiemethode geïmplementeerd, en wij geven een complexiteitsanalyse van het bijbehorende algoritme. Het verband tussen Mealy machines en rationale specificaties wordt ook verder onderzocht. Wij geven een bovengrens op het aantal toestanden van de geconstrueerde Mealy

machine in termen van de parameters van de specificatie. Ook laten wij zien dat niet alle eindige Mealy machines de realisatie is van een rationale specificatie.

In Hoofdstuk 4 richten wij ons op het coalgebraïsch modelleren van subsequentiële transducers. Subsequentiële transducers hebben toepassingen binnen de coderingstheorie en de taalverwerking. Ook vormen zij een generalisatie van Mealy machines en klassieke automaten, die allebei een nette coalgebraïsche modellering toestaan. De structuur van een subsequentiële transducer ziet er ook uit als een coalgebra, maar het blijkt vrij eenvoudig in te zien dat de bijbehorende definitie van coalgebra morfisme niet goed past bij de semantiek van subsequentiële transducers. Wij laten zien dat een juiste coalgebraïsche modellering van genormaliseerde subsequentiële transducers wel mogelijk is, en dat een finale genormaliseerde subsequentiële transducer bestaat. Dit resultaat biedt een alternatief perspectief op bepaalde resultaten uit de bestaande theorie. Het verklaart onder andere het feit dat subsequentiële transducers geminimaliseerd kunnen worden door eerst te normaliseren en daarna alle bisimilaire toestanden te identificeren. Wij laten in dit hoofdstuk ook zien dat normalisatie, minimalisatie en het nemen van differentialen reflectoren zijn in de categorie-theoretische zin.

In Hoofdstuk 5 nemen wij als uitgangspunt de coalgebraïsche modellering van omgevingsstructuren (Engels: 'neighbourhood structures'). Omgevingsstructuren vormen een generalisatie van Kripke-structuren, en de bijbehorende niet-normale modale logica wordt 'klassieke modale logica' genoemd. Wij gebruiken eerst de coalgebraïsche modellering om drie noties van semantische equivalentie te definiëreen. Een daarvan is een nieuwe coalgebraïsche notie die in ligt tussen bisimulatie en gedragsequivalentie. In de modale logica van Kripke-modellen is het verband tussen bisimilariteit en de uitdrukkingskract van de modale taal bekend uit resultaten zoals de Hennessy-Milner stelling en Van Benthem's karakterizeringsstelling. Wij bewijzen hier analoga van deze resultaten voor klassieke modale logica, en geven ook een model-theoretisch bewijs van Craig-interpolatie, waarin coalgebraïsche bisimulaties een belangrijke rol spelen.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue*. Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms*. Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications*. Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures*. Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability*. Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions*. Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments*. Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression*. Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages*. Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

T. Gelsema. *Effective Models for the Structure of pi-Calculus Processes*

with Replication. Faculty of Mathematics and Natural Sciences, UL. 2005-17

P. Zoeteweij. *Composing Constraint Solvers*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-18

J.J. Vinju. *Analysis and Transformation of Source Code by Parsing and Rewriting*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2005-19

M.Valero Espada. *Modal Abstraction and Replication of Processes with Data*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2005-20

A. Dijkstra. *Stepping through Haskell*. Faculty of Science, UU. 2005-21

Y.W. Law. *Key management and link-layer security of wireless sensor networks: energy-efficient attack and defense*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-22

E. Dolstra. *The Purely Functional Software Deployment Model*. Faculty of Science, UU. 2006-01

R.J. Corin. *Analysis Models for Security Protocols*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-02

P.R.A. Verbaan. *The Computational Complexity of Evolving Systems*. Faculty of Science, UU. 2006-03

K.L. Man and R.R.H. Schiffelers. *Formal Specification and Analysis of Hybrid Systems*. Faculty of Mathematics and Computer Science and Faculty of Mechanical Engineering, TU/e. 2006-04

M. Kyas. *Verifying OCL Specifications of UML Models: Tool Support and Compositionality*. Faculty of Mathematics and Natural Sciences, UL. 2006-05

M. Hendriks. *Model Checking Timed Automata - Techniques and Applications*. Faculty of Science, Mathematics and Computer Science, RU. 2006-06

J. Ketema. *Böhm-Like Trees for Rewriting*. Faculty of Sciences, VUA. 2006-07

C.-B. Breunesse. *On JML: topics in tool-assisted verification of JML programs*. Faculty of Science, Mathematics and Computer Science, RU. 2006-08

B. Markvoort. *Towards Hybrid Molecular Simulations*. Faculty of Biomedical Engineering, TU/e. 2006-09

S.G.R. Nijssen. *Mining Structured Data*. Faculty of Mathematics and Natural Sciences, UL. 2006-10

G. Russello. *Separation and Adaptation of Concerns in a Shared Data Space*. Faculty of Mathematics and Computer Science, TU/e. 2006-11

L. Cheung. *Reconciling Nondeterministic and Probabilistic Choices*. Faculty of Science, Mathematics and Computer Science, RU. 2006-12

B. Badban. *Verification techniques for Extensions of Equality Logic*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2006-13

A.J. Mooij. *Constructive formal methods and protocol standardization*. Faculty of Mathematics and Computer Science, TU/e. 2006-14

T. Krilavicius. *Hybrid Techniques for Hybrid Systems*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-15

M.E. Warnier. *Language Based Security for Java and JML*. Faculty of Science, Mathematics and Computer Science, RU. 2006-16

V. Sundramoorthy. *At Home In Service Discovery*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2006-17

B. Gebremichael. *Expressivity of Timed Automata Models*. Faculty of Science, Mathematics and Computer Science, RU. 2006-18

L.C.M. van Gool. *Formalising Interface Specifications*. Faculty of Mathematics and Computer Science, TU/e. 2006-19

C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Faculty of Mathematics and Computer Science, TU/e. 2006-20

J.V. Guillen Scholten. *Mobile Channels for Exogenous Coordination of Distributed Systems: Semantics, Implementation and Composition*. Faculty of Mathematics and Natural Sciences, UL. 2006-21

H.A. de Jong. *Flexible Heterogeneous Software Systems*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-01

N.K. Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-02

M. van Veelen. *Considerations on Modeling for Early Detection of Abnormalities in Locally Autonomous Distributed Systems*. Faculty of Mathematics and Computing Sciences, RUG. 2007-03

T.D. Vu. *Semantics and Applications of Process and Program Algebra*. Faculty of Natural Sciences, Mathematics, and Computer Science, UvA. 2007-04

L. Brandán Briones. *Theories for Model-based Testing: Real-time and Coverage*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-05

I. Loeb. *Natural Deduction: Sharing by Presentation*. Faculty of Science, Mathematics and Computer Science, RU. 2007-06

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical

Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems*. Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML*. Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery*. Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding*. Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks*. Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data*. Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy*. Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS*. Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03

A.M. Marin. *An Integrated System to Manage Crosscutting Concerns in Source Code*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-04

N.C.W.M. Braspenning. *Model-based Integration and Testing of High-tech Multi-disciplinary Systems*. Faculty of Mechanical Engineering, TU/e. 2008-05

M. Bravenboer. *Exercises in Free Syntax: Syntax Definition, Parsing, and Assimilation of Language Conglomerates*. Faculty of Science, UU. 2008-06

M. Torabi Dashti. *Keeping Fairness Alive: Design and Formal Ver-*

ification of Optimistic Fair Exchange Protocols. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2008-07

I.S.M. de Jong. *Integration and Test Strategies for Complex Manufacturing Machines*. Faculty of Mechanical Engineering, TU/e. 2008-08

I. Hasuo. *Tracing Anonymity with Coalgebras*. Faculty of Science, Mathematics and Computer Science, RU. 2008-09

L.G.W.A. Cleophas. *Tree Algorithms: Two Taxonomies and a Toolkit*. Faculty of Mathematics and Computer Science, TU/e. 2008-10

I.S. Zapreev. *Model Checking Markov Chains: Techniques and Tools*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-11

M. Farshi. *A Theoretical and Experimental Study of Geometric Networks*. Faculty of Mathematics and Computer Science, TU/e. 2008-12

G. Gulesir. *Evolvable Behavior Specifications Using Context-Sensitive Wildcards*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-13

F.D. Garcia. *Formal and Computational Cryptography: Protocols, Hashes and Commitments*. Faculty of Science, Mathematics and Computer Science, RU. 2008-14

P. E. A. Dürr. *Resource-based Verification for Robust Composition of Aspects*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-15

E.M. Bortnik. *Formal Methods in Support of SMC Design*. Faculty of Mechanical Engineering, TU/e. 2008-16

R.H. Mak. *Design and Performance Analysis of Data-Independent Stream Processing Systems*. Faculty of Mathematics and Computer Science, TU/e. 2008-17

M. van der Horst. *Scalable Block Processing Algorithms*. Faculty of Mathematics and Computer Science, TU/e. 2008-18

C.M. Gray. *Algorithms for Fat Objects: Decompositions and Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-19

J.R. Calamé. *Testing Reactive Systems with Data - Enumerative Methods and Constraint Solving*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-20

E. Mumford. *Drawing Graphs for Cartographic Applications*. Faculty of Mathematics and Computer Science, TU/e. 2008-21

E.H. de Graaf. *Mining Semi-structured Data, Theoretical and Experimental Aspects of Pattern Evaluation*. Faculty of Mathematics and Natural Sciences, UL. 2008-22

R. Brijder. *Models of Natural Computation: Gene Assembly and Membrane Systems*. Faculty of Mathematics and Natural Sciences, UL. 2008-23

A. Koprowski. *Termination of Rewriting and Its Certification.* Faculty of Mathematics and Computer Science, TU/e. 2008-24

U. Khadim. *Process Algebras for Hybrid Systems: Comparison and Development.* Faculty of Mathematics and Computer Science, TU/e. 2008-25

J. Markovski. *Real and Stochastic Time in Process Algebras for Performance Evaluation.* Faculty of Mathematics and Computer Science, TU/e. 2008-26

H. Kastenberg. *Graph-Based Software Specification and Verification.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-27

I.R. Buhan. *Cryptographic Keys from Noisy Data Theory and Applications.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-28

R.S. Marin-Perianu. *Wireless Sensor Networks in Motion: Clustering Algorithms for Service Discovery and Provisioning.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2008-29

M.H.G. Verhoef. *Modeling and Validating Distributed Embedded Real-Time Control Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2009-01

M. de Mol. *Reasoning about Functional Programs: Sparkle, a proof assistant for Clean.* Faculty of Science, Mathematics and Computer Science, RU. 2009-02

M. Lormans. *Managing Requirements Evolution.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2009-03

M.P.W.J. van Osch. *Automated Model-based Testing of Hybrid Systems.* Faculty of Mathematics and Computer Science, TU/e. 2009-04

H. Sozer. *Architecting Fault-Tolerant Software Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2009-05

M.J. van Weerdenburg. *Efficient Rewriting Techniques.* Faculty of Mathematics and Computer Science, TU/e. 2009-06

H.H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic.* Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2009-07