Canonical Algebraic Generators in Automata Learning

Stefan Jens Zetzsche

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
at
University College London
(UCL)

Department of Computer Science

August, 2023

Declaration

I, Stefan Jens Zetzsche confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Many methods for the verification of complex computer systems require the existence of a tractable mathematical abstraction of the system, often in the form of an automaton. In reality, however, such a model is hard to come up with, in particular manually. Automata learning is a technique that can *automatically* infer an automaton model from a system – by observing its behaviour. The majority of automata learning algorithms is based on the so-called L^* algorithm. The acceptor learned by L^* has an important property: it is *canonical*, in the sense that, it is, up to isomorphism, the *unique* deterministic finite automaton of *minimal size* accepting a given regular language. Establishing a similar result for other classes of acceptors, often with side-effects, is of great practical importance. Non-deterministic finite automata, for instance, can be exponentially more succinct than deterministic ones, allowing verification to scale. Unfortunately, identifying a canonical size-minimal *non-deterministic* acceptor of a given regular language is in general not possible: it can happen that a regular language is accepted by two *non-isomorphic* non-deterministic finite automata of minimal size. In particular, it thus is unclear which one of the automata should be targeted by a learning algorithm. In this thesis, we further explore the issue and identify (sub-)classes of acceptors that admit canonical size-minimal representatives.

In more detail, the contributions of this thesis are three-fold.

First, we expand the automata (learning) theory of Guarded Kleene Algebra with Tests (GKAT), an efficiently decidable logic expressive enough to model simple imperative programs. In particular, we present GL^*, an algorithm that learns the unique size-minimal GKAT automaton for a given deterministic language, and prove that GL^* is more efficient than an existing variation of L^*. We implement both algorithms in OCaml, and compare them on example programs.

Second, we present a category-theoretical framework based on generators, bialgebras, and distributive laws, which identifies, for a wide class of automata with side-effects in a monad, canonical target models for automata learning. Apart from recovering examples from the literature, we discover a new canonical acceptor of regular languages, and present a unifying minimality result.

Finally, we show that the construction underlying our framework is an instance of a more general theory. First, we see that deriving a minimal bialgebra from a minimal coalgebra can be realized by applying a monad on a category of subobjects with respect to an epi-mono factorisation system. Second, we explore the abstract theory of generators and bases for algebras over a monad: we discuss bases for bialgebras, the product of bases, generalise the representation theory of linear maps, and compare our ideas to a coalgebra-based approach.

Impact

Outside Academia As hardware and software systems continue to grow in complexity, methods for their verification become increasingly important. Classical model checking approaches to verification require the prior existence of a rich model of the system of interest, able to express all its relevant behaviour. In reality such a model is often unavailable, for instance, when the system comes in the form of a black-box with no access to the source code, or the system is simply too complex for manual processing. The black-box automata learning technique addresses this issue and has been successfully applied in a wide range of use cases, from finding bugs in network protocols [48], reverse engineering smartcard reader for internet banking [42], and other industrial applications [61]. A comprehensive survey can be found in [153].

One of the bottle-necks for learning algorithms in an industrial setting is scalability. Identifying canonical target acceptors of minimal size is thus of great practical importance. As it happens, establishing uniqueness and minimality results for classes of acceptors with side-effects, which are often exponentially more succinct than their deterministic counterparts, can be surprisingly difficult [49]. In this thesis, we provide a general categorical framework that incorporates existing constructions of canonical minimal automata with side-effects and unveils new ones, and present a unifying minimality result. Another variable that impacts scalability is the maximum number of observation and equivalence queries an algorithm requires to learn a model. In this thesis, we present GL^*, which deduces automata representations of simple imperative programs, and generally requires less queries than existing approaches. This is particularly interesting in view of potential applications to network verification [143].

Inside Academia The lack of a canonical target acceptor when learning non-deterministic models has lead to a number of independent approaches, for different variants of non-determinism [28, 53]. More recently, there have been efforts to give a unifying perspective on those approaches. In this thesis, we present a category-theoretic framework that generalises ideas of Van Heerdt [63, 64, 66], the notion of a scoop by Arbib and Manes [19], and the universal-algebraic treatment of Myers et al. [119]. We present the first general algorithm for the construction of succinct automata and give a unifying minimality result. In particular, we discover a previously unknown canonical acceptor of regular languages. By using bialgebras and distributive laws, we are able to clarify the central role of algebraic generators and bases, which has previously been underappreciated. We further explore this direction by developing the independent abstract theory of generators and bases. We expect those results to be also valuable outside of the automata learning community.

Guarded Kleene Algebra with Tests is a variation of Kleene Algebra with Tests that one obtains by restricting the union and iteration operations to guarded versions [142]. Recently, this variation has become the subject of increasing interest [135]. We contribute to the development of its automata (learning) theory, for example, by establishing the existence of a unique size-minimal acceptor. Our treatment leads to numerous directions that could be further explored; for instance improving efficiency through more compact data-structures, or an adaption to a probabilistic extension.

The results of this thesis have been published at the 37th and 38th International Conference on Mathematical Foundations of Programming Semantics [161, 160], and at the 10th Conference on Algebra and Coalgebra in Computer Science [159]. Additionally, some of the results have been presented at the 8th Symposium on Compositional Structures [161].

Acknowledgements

First of all, my thanks go to my two supervisors, Alexandra and Matteo. I am grateful for you giving me the chance to pursue a PhD in the first place. Studying abroad, in London, has been something I have always wanted to do. At all times, you two have been patient and encouraging. Thank you for supporting and guiding me through good and bad times. Alexandra, I have learnt a lot from you. Your sharp intellect, good heart, and work ethic will stay with me. I will also remember your delicious cooking. Matteo, you have always had the right advice for me. Your writing has been as beautiful and simple as the food of Italy. I will miss you two.

I would also like to thank all the peers that have taken the time to review my work and provide valuable feedback. Over time, I received many insightful comments on my submissions, often from anonymous reviewers. My coauthors Alex, Gerco, and Matteo have given me fruitful guidance during numerous discussions. I am particularly grateful to Mehrnoosh and Stefan, for agreeing to be my examiners, and for studying my thesis in thorough detail. I am also thankful to Fabio and James, who chaired my transfer and first year viva, respectively. Both experiences were challenging, but helped me a lot with progressing towards an independent researcher.

I am very fortunate to have been a part of the Programming Principles, Logic, and Verification group. All the numerous seminars, reading groups, and informal conversations were a welcome change to my daily routine. Over the years I shared my office with many inspiring people. My first desk has been in a room with Gerco, Jana, Paul, and Tobias, all of which are greatly missed. Later, I moved to the basement, which I shared with Jas, Leo, Linpeng, Louis, Mateo, Robin, Tiago, Will, and Wojciech. The collaborative spirit in our department allowed me to converse with researchers of all experiences. Thank you, Bas, Benjamin, Christoph, Diana,

Fredrik, Jurriaan, Lachlan, Maria, Paul, Simon, Sonia, Tao, Todd, and everyone else that made our group so unique.

During my PhD I was able to attend, volunteer, and mentor at various conferences and schools, first in person, and later online. Among others, I am grateful to the organisers of the following events: Syco Birmingham 2018, Syco Strathclyde 2018, VeTSS Workshop on Formal Methods and Tools for Security 2018, Calco 2019, Facebook Proofs for Bugs 2019, Mfps 2019, Scottish Programming Languages and Verification Summer School 2019, VeTSS Verified Software Workshop 2019, Pldi 2020, Popl 2020, Splash 2020, Cav 2021, Icalp 2021, Mfps 2021, Popl 2021, Syco Tallinn 2021, Cav 2022, Mfps 2022. One of the highlights has been the trip to Popl in New Orleans, in early 2020, just before the first Covid cases became public. I remember exploring the town with Jana and Tobias, and dancing to live music, late at night.

In addition to my own studies, I have been working as a Teaching Assistant for a number of courses. I would recommend everyone to do the same during their PhD; it is a humbling experience. Over the years, I was lucky to meet many students, often bright, always unique in character. For some courses, I collaborated with other PhD students, which was particularly fun. Thank you, Tao, Thomas, and Todd!

I was fortunate enough to experience academia not only from the inside, but also from the outside. Towards the end of my PhD, I completed two twelve-week internships in the tech industry: first at Amazon, then at Meta. Both experiences have been very rewarding and have greatly broadened my horizon. It would be a hopeless attempt to name all of the people I met along the way. There are, however, a few people I would like to thank particularly.

First of all, there is Byron, who, when I hesitantly reached out to him with the idea of doing an Amazon internship, replied, to my surprise, instantly, and encouraged me in his unique way. Thank you, Byron, for forwarding me to the right places.

During the internship I was supervised by Rustan, who was based in Seattle, while I was working from London. I could not have asked for a better match. Rustan was very generous. He took the time to meet me virtually on a daily basis and has answered my endless questions in impressive clarity and depth, and with great spirit. On top of all, Rustan turned out to be also a tremendous chef and host.

During my time in the London office I met many great people. Thank you Car-

oline, Claudia, Daniel, Ilina, Sacha, and everyone else. You have made working remotely so much easier for me.

As Covid cases were coming down, I was able to complete my second internship at Meta in person, in London. The Hack language team has been more than welcoming. Thank you Andrew, Frank, Henri, Max, Michael, Mistral, Scott, and everyone else I met along the way. Our trips to Cambridge and Menlo Park have been highlights I'll cherish. In particular I would like to thank Mistral, my supervisor, who has been a great mentor and friend. I miss our coffee runs and trips to the climbing hall.

Around the end of 2020, I applied to the SIGPLAN long-term mentorship program. I am happy to say that since then I have met my mentor on an almost bi-weekly basis. Thank you, Ravi, your support and sincerity mean a lot to me. I hope we manage to see each other in person in the near future.

Thanks also to my friends and family for their continuous support during this long journey. In particular, to my parents, Frank and Gabriele, for their unconditional love over all the years. Without you two this thesis would not have been possible.

Finally, I'd like to say thanks to Aurora. Thank you for your love, advice, and understanding. For helping me through the lows, and for celebrating the highs. You have played a great part in the success of this thesis.

[0]My research has been supported by GCHQ via the VeTSS grant *Automated Black-Box Verification of Networking Systems* (4207703/RFA 15845) and by the ERC via the Consolidator Grant *AutoProbe* (101002697).

Contents

Chapter 1

Introduction

As hardware and software systems continue to grow in size and complexity, methods for their analysis become increasingly important. To study the characteristics of a system, classical approaches require the existence of a simplified mathematical model that captures the relevant behaviour of the system. One of the simplest models is the *automaton*, which can be thought of as a diagram generated by the states of the system and transitions between them. Unfortunately, in reality a complex model is only rarely available, for instance, when the system comes in the form of a black-box with no access to the source code, or the system is too complex for manual processing.

The aim of *automata learning* is to automatically infer a minimal sized automata representation of a system by observing its behaviour. The incremental approach has been successfully applied to a wide range of verification tasks from finding bugs in network protocols [48], reverse engineering smartcard reader for internet banking [42], and industrial applications [61]. A comprehensive survey of the key developments in automata learning can be found in [153]. As a result of its success, automata learning has inspired numerous adaptions which target more expressive models.

This thesis develops along two orthogonal axes. On the one hand, we contribute to the branches of automata learning by developing an algorithm that efficiently learns a model that captures the behaviour of a simple imperative program by applying domain-specific optimisations. On the other hand, we further develop the abstract perspective on automata learning by presenting a mathematical framework that unifies numerous constructions of minimal target models, and unveils new ones.

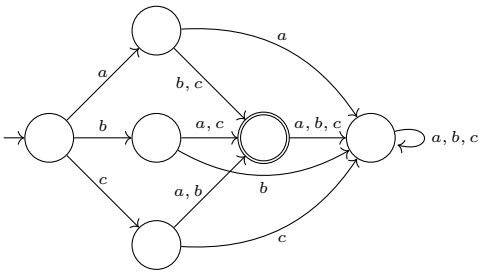


Figure 1.1: Up to isomorphism, the unique size-minimal DFA accepting the language $\{ab, ac, ba, bc, ca, cb\} \subseteq \{a, b, c\}^*$ [20]

The introduction is structured as follows. In Section 1.1 we present the seminal L^* automata learning algorithm. We continue with an example run of L^* in Section 1.2. The content of Section 1.3 is an overview of adaptions of L^* to other types of target models. In Section 1.4 we present Guarded Kleene Algebra with Tests, and discuss its ramifications with automata learning. We then discuss the concept of size-minimality for target models in Section 1.5. The unifying perspective of category theory on automata learning is explored in Section 1.6. We continue with a presentation of the main objectives of the thesis in Section 1.7, and conclude with an overview of the structure and the contributions of the thesis in Section 1.8.

1.1 Automata Learning

Automata learning can be broadly divided into *active* and *passive* learning algorithms. Active learning algorithms infer a model from a system by interacting with (or *querying*) the system during the execution of the algorithm. In contrast, during passive learning, there is no direct access to a system for the duration of the run of an algorithm. Instead, passive learning algorithms try to model a system based on potentially insufficient stored data obtained through prior observation. In this thesis, we will focus on active automata learning.

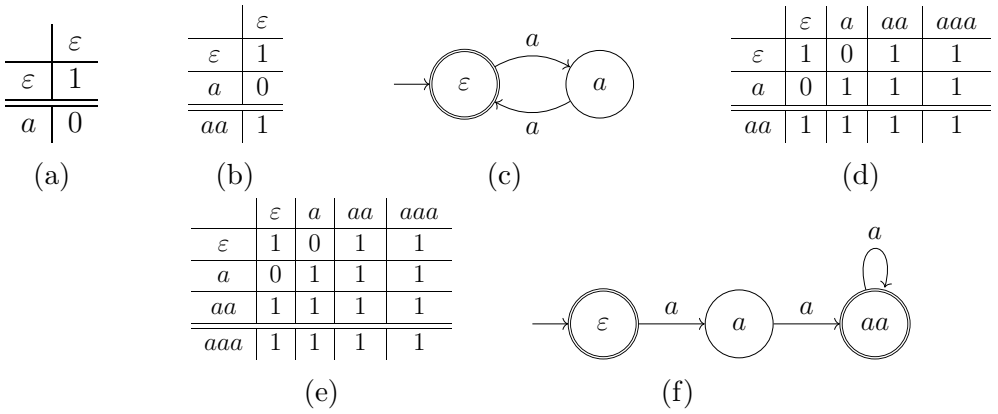
A *non-deterministic finite automaton* (NFA) over a fixed input alphabet set of characters consists of a set of states, partitioned into *accepting* and *rejecting* states, a distinguished initial state, and a transition function that assigns to each state and input character a *set* of next states. If at every transition the set of next states

consists of a single state, then we speak of a *deterministic finite automaton* (DFA). Any NFA can be depicted as a directed graph of nodes, which represent its states, and arrows, which represent transitions, thus are labelled by input characters. The node representing the initial state is annotated by an arrow without domain. Nodes that correspond to accepting states are indicated by a double circle. A simple example of a DFA over the input alphabet $\{a, b, c\}$ with six states of which one is accepting is given in Figure 1.1.

A finite sequence of characters in the input alphabet is referred to as *word*. A set of words is called a *language* (over the input alphabet). A word is *accepted* by a DFA, if consecutively reading in its characters leads to a transition from the initial state to an accepting state. A DFA *accepts* the language that consists of the words it accepts. A language that is accepted by a DFA is called *regular*. For any regular language there is a unique size-minimal DFA accepting it, defined up to a structure-preserving bijection. For example, the size-minimal DFA accepting the regular language of words over the alphabet $\{a, b, c\}$ that have length two and start and end in different characters is depicted in Figure 1.1.

In active automata learning it is usually assumed that the behaviour of the system one tries to model is given in terms of an unknown regular language. In consequence, there exists, in principle, an (unknown) size-minimal DFA that accurately models the behaviour of the system. Under this assumption, the most simple form of interacting with a system is through a *membership query*. During such an idealised interaction, the learning algorithm *submits* a word to the system and observes its response. Either, the word is *accepted*, that is, it is an element of the regular language that represents the behaviour of the system, or, it is *rejected*.

In [118] Moore showed that membership queries alone are insufficient to deduce, in finite time, a DFA that correctly models the behaviour of a system. Later, Angluin [13] proved that with the auxiliary knowledge of the number n of states of the minimal DFA accepting the behaviour of the system, there exists a correct algorithm that converges in finite time, but performs a number of membership queries that is exponential in n in the worst case. Consequently, Angluin described and studied several other types of queries, in particular, the *equivalance query* [15]. During such a query, a hypothesis model is submitted, and either accepted, if it correctly models the target



(a) (b) (c) (d) (e) (f)

Figure 1.2: An example run of (a variation of) Angluin's L^* algorithm for the target regular language $1 + a \cdot a \cdot a^* = \{\varepsilon, aa, aaa, ...\} \subseteq \{a\}^*$

behaviour, or rejected. In the case of rejection, a *counterexample* is provided, that is, a word that is either incorrectly accepted or incorrectly rejected by the hypothesis.

In her seminal work [14] Angluin presented L^*, an algorithm that learns the minimal DFA accepting a target regular language, by assuming a *minimally adequate teacher*, which can answer membership and equivalence queries. The algorithm runs in time polynomial in the number of states of the minimal DFA and the maximum length of any counterexample.

The minimal adequate teacher framework should be understood as an elegant mathematical abstraction, rather than a precise implementation guideline. For instance, verifying whether the behaviour of the hypothesis matches the target language can in practice easily become unfeasible. Equivalence queries are thus often approximated via testing [43]. If a bound to the size of the target model is known, or the target language is represented by a known automaton for experimental purposes, equivalence queries can be performed exactly. In the latter case, for example, one can utilise efficient bisimulation algorithms (see e.g. [70, 32]).

1.2 An Example Run of L^*

We will now give an intuitive account of how Angluin's L^* algorithm can be used to learn via membership and equivalence queries the minimal DFA accepting a given

regular language. In our example, we fix the regular target language $L = 1 + a \cdot a \cdot a^*$, which consists of words over the singleton input alphabet $\{a\}$, either of length 0 or of length at least 2. The complete run of L^* (more precisely, a slight variation[1] of it) for L can be found in Figure 1.2. Its result, the minimal DFA accepting L, has 3 states, and is depicted in Figure 1.2f.

At the heart of L^* is a data-structure called *observation table*, which consists of partial information about L, gathered by performing membership queries. The binary value of an observation table at row i and column j specifies whether the word $w_i \cdot w_j$, obtained by concatenating the words w_i and w_j at respective indices, is contained in L, or not. In the former case, the table contains a one, and in the latter case a zero. An observation table consists of two disjoint parts: an upper part, and a lower part, visually distinguished by two horizontal lines. Every row in the upper part of the table denotes the state of a hypothesis automaton that is not necessarily well-defined yet. The lower part of the table is used to establish the transitional structure of the automaton. During the run of L^*, the observation table is extended until a well-defined hypothesis can be constructed. The hypothesis is then checked for equivalence, leading to termination, if positive, or further refinement of the table via a counterexample, if negative.

Initially, the observation table for L is initiated with one upper row and one column, both indexed by the empty word ε. Since the concatenation $\varepsilon = \varepsilon \cdot \varepsilon$ is accepted by L, the upper left entry of the table in Figure 1.2a contains a one. To deduce from the table an automaton, we need to determine the state that is reached from the state indexed by ε, when reading in the character a. To do so, we construct the lower row indexed by $a = \varepsilon \cdot a$. The resulting table is depicted in Figure 1.2a.

Since the row indexed by a differs from all rows in the upper part of the table (that is, the one indexed by ε), the automaton potentially corresponding to the table in Figure 1.2a contains at least two states. Formally, we move the row indexed by a to the upper part of the table. In consequence, we also need to determine the state that is reached from the state indexed by a, when reading in the character a. The

[1] In contrast to the original presentation of L^*, which adds rows for all prefixes of a counterexample, we use a variation by Maler and Pnueli [108], which adds columns for all suffixes of a counterexample. This has the advantage that *consistency* checks become redundant.

new table, with a row indexed by $a \cdot a$ in its lower part, is depicted in Figure 1.2b. As every row in the lower part of the table coincides with one upper row, we now can deduce a well-defined automaton.

The well-defined hypothesis corresponding to the observation table in Figure 1.2b is given in Figure 1.2c. Its initial state corresponds to the row indexed by ε. The initial state is also accepting, because the corresponding row contains a one at the column indexed by ε. Is the behaviour of the hypothesis given by the target language L? There exists at least one counterexample that witnesses that this is not the case. For example, the word aaa is accepted by L, but rejected by the hypothesis.

To account for the imprecise behaviour, we add to the observation table in Figure 1.2b one column for each suffix of the counterexample aaa. The updated table is depicted in Figure 1.2d. While the rows indexed by ε and aa have previously contained identical values, they now differ, as is witnessed by their entry at the column indexed by a. As the lower row indexed by aa now differs from all upper rows, we move it to the upper part of the table. To derive a well-defined transitional structure, we add a new row indexed by aaa to the lower part of the table. The resulting structure is depicted in Figure 1.2e.

Since every row in the lower part of the table corresponds to one row in the upper part of the table, we can derive the well-defined hypothesis Figure 1.2f. As one verifies, the language accepted by the hypothesis is precisely the target language L, which completes the execution of L^*.

1.3 Other Types of Models

Since its publication, Angluin's seminal L^* algorithm [14] for learning the minimal DFA accepting a given regular language has inspired numerous variations. On the one hand, authors have adjusted L^* by using more efficient data structures [83, 74] and handling counterexamples differently [130]. On the other hand, L^* has been extended to output models other than DFAs, often either equally expressive, but more succinct, or more expressive, but still efficiently learnable. In this section, we will focus on the latter type, and give a few examples of such target models.

Some authors have remarked that DFAs are inappropriate to capture the be-

haviour of many complex systems [139]. This is due to the behaviour of such systems being often naturally characterized in terms of complex input-output pairs: the system receives an input from the environment, transitions, and produces an output to the environment. DFAs lack such general input-output behaviour, being classifiers, which either output 0 (reject) or 1 (accept).

A more natural model for systems exhibiting general input-output behaviour are *Mealy machines* [109], which often are also more concise than DFAs. Learning algorithms for Mealy machines based on L^* have appeared in [124, 139]. Practical introductions to the general development of active learning, with a focus on Mealy machines, were given in [145, 144].

Mealy machines are as expressive as *Moore automata* [118], which generalise DFA from the two-element Boolean set to an arbitrary output set. Angluin's L^* algorithm can naturally be extended to Moore automata. An optimized version for learning the products of Moore automata has been presented in [110]. A passive learning algorithm for Moore automata is the subject of [59].

Another class of automata exhibiting complex input-output behaviour are *weighted automata*, which generalise NFAs from the two-element Boolean semiring to arbitrary semirings. Weighted automata have been used in text processing [117], character recognition [37], image processing [8, 45], bioinformatics [9], and formal verification [11]. The characteristics of weighted automata over arbitrary semirings have been extensively studied [134, 116]. An active-learning algorithm for weighted automata over the field of rationals inspired by L^* appeared first in [26], and has later been generalized to weighted automata over arbitrary fields [27]. A passive learning algorithm is the subject of [23]. A survey of the developments until 2015 can be found in [22]. The active learning of weighted automata over general semirings is explored in [68].

In many situations automata dealing with infinite instead of finite structures are a more natural and realistic model of the behaviour of a system. We would like to emphasise two particular classes of such models. First, automata that allow infinite input alphabets (typically equipped with rich additional structure), while characterising words of finite length. Second, automata that require finite input alphabets, but characterise words of infinite length.

Three examples of the former type are *register automata*, *symbolic automata*, and

nominal automata.

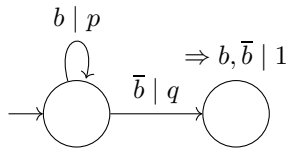
Register automata (originally called *finite-memory automata* [82]) extend deterministic automata to infinite input alphabets by introducing a register that enables the storage of data for future comparison [51]. Active learning algorithms for register automata have been the subject of numerous publications [72, 30, 3, 41]. A review of the developments was given in [73, 2].

Symbolic automata are automata with transitions labelled by predicates over an input alphabet that is a Boolean algebra of possibly infinite size. The theoretical aspects of symbolic automata have been extensively studied [148]. A minimisation procedure, for example, appears in [47]. The learnability of symbolic automata *in the limit* is the content of [56]. Angluin-style learning algorithms for symbolic automata appear in [107, 50].

Nominal sets are sets that are finitely supported with respect to a group action of permutations on a countably infinite set [125]. Originally introduced as an alternative to set theory, they have been later rediscovered for name binding in the context of programming languages. Nominal automata appear, among others, in [97, 137]. Essentially, they generalise DFA to orbit-finite nominal sets and equivariant functions. An adaptation of L^* to nominal automata is the content of [112, 111].

An example of the latter type are *Büchi automata*, which accept the ω-regular languages of words of infinite length. Büchi automata are central to the automata-based model checking approach to verification [155]. As such, they have been used to describe properties of distributed systems [10], are relevant to the synthesis of reactive systems [126], and appear in termination proofs for programs [98]. The first learning algorithm for Büchi automata accepting a strict subclass of ω-regular languages has been introduced by Maler and Pnueli [108]. The first learning algorithm accepting the full class of ω-regular languages has appeared in [54], and is based on ideas in [14] and [40]. Another learning algorithm accepting the complete class of ω-regular languages appeared in [102, 103]. Among others, it introduced a more efficient data structure [83, 74] to the work of Angluin and Fisman [18].

Apart of above cases, L^* has also been extended to non-deterministic finite automata [31], universal finite automata [17], and alternating finite automata [17, 28], all of which are as expressive as DFAs.



(while b do p); q (a) e (b) $\mathscr{X}_e : X \to (2 + \Sigma \times X)^{\mathsf{At}}$ (c) $\{\overline{b}qb, \overline{b}q\overline{b}, bp\overline{b}qb, bp\overline{b}q\overline{b}, ...\}$ $[\![e]\!] = [\![\mathscr{X}_e]\!] \subseteq (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$

Figure 1.3: The interplay between expressions, automata, and languages in GKAT, for $\Sigma = \{p, q\}$ and $\mathsf{At} = \{b, \overline{b}\}$

1.4 Guarded Kleene Algebra with Tests

Even though there are already numerous extensions of Angluin's seminal L^* algorithm, for many different types of target models, one can still find interesting unexplored domains for its potential application. One such domain is *Guarded Kleene Algebra with Tests* (GKAT), a logic that has recently become the subject of increasing interest [142, 135]. This attention stems from a few remarkable characteristics.

First, it is based on *Kleene Algebra with Tests*, a well-studied logic with sound mathematical foundations [92, 89]. Through this relationship many constructions for it are either directly induced, or at least hinted at. Second, the behaviour of GKAT expressions $e, f, ...$ can be identified with the one of simple imperative programs (cf. Figure 3.3 on page 65). It is, for instance, possible to iterate expressions, $e; f$, or build new expression via program-flow constructions such as `if` b `then` e `else` f and `while` b `do` e. Third, while remaining overall sufficiently expressive, equivalence of expressions in GKAT is more efficiently decidable than in its foundations. This makes the logic particular attractive for scale-sensitive applications such as network verification [143].

For black-box learning in the spirit of L^*, GKAT is particularly well suited, because of its well-behaved interplay of expressions, automata, and languages, that closely resembles the one of regular expressions, DFAs, and regular languages. In Figure 1.3c, for example, is depicted the language which models the behaviour of both the GKAT expression (`while` b `do` p); q, and the GKAT automaton in Figure 1.3b. More generally, one can, for any GKAT expression, efficiently construct a GKAT automaton that accepts the same language, and, reversely, for every automaton of a particular kind, one can find a language equivalent expression (for details see [142]).

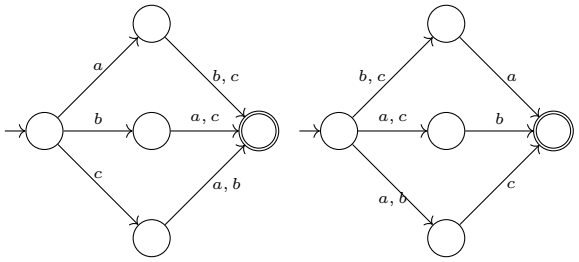


Figure 1.4: Two non-isomorphic size-minimal NFA accepting the language $\{ab, ac, ba, bc, ca, cb\} \subseteq \{a, b, c\}^*$ [20]

In this thesis we investigate how the ideas behind L^* can be used to derive, from program traces, behavioural equivalent GKAT automata representations, that is, simple abstractions of imperative programs. Having GKAT's potential applications in mind, we are particularly interested in domain-specific optimisations that reduce the number of involved membership queries, and ways to construct automata representations that are as compact as possible.

1.5 Size-Minimality

The deterministic finite-state automaton for a given regular target language derived by L^* has a remarkable property: first, it accepts the target language, and second, every other deterministic finite-state automaton accepting the target language has either a state space of greater size, or is equivalent up to structure-preserving bijection. In other words, L^* learns a *canonical* representation: the unique size-minimal DFA.

As is well-known, the canonical representation of a regular language as DFA can be explicitly constructed, via the *Myhill-Nerode relation* [120]. Under this construction, states of the minimal DFA for a language $L \subseteq A^*$ over an input alphabet A can be identified with equivalence classes of *residual languages* of the type $w^{-1}L = \{w{\cdot}u \mid u \in A^*\}$, for $w \in A^*$. It is not hard to see that the observation table data-structure of L^* closely resembles this construction. Indeed, at every step, rows indexed by a word $w \in A^*$ approximate the residual language $w^{-1}L$, since their entry at a column indexed by $u \in A^*$ coincides with the evaluation $u \in w^{-1}L$. This illustrates that the design of a learning algorithm should start with the identification and explicit construction

(a) The size-minimal DFA (b) The canonical RFSA

Figure 1.5: Two canonical acceptors for $(a + b)^*a$

of a *canonical* target model. The algorithm itself is then merely a derivation.

Unfortunately, not all classes of acceptors admit a canonical representative. For example, while there exists, up to isomorphism, precisely one size-minimal DFA accepting the regular language over the alphabet $\{a, b, c\}$ of words of length two, starting and ending in different characters (Figure 1.1), there exist at least two size-minimal NFAs that are non-isomorphic (Figure 1.4). This immediately leads to the question: what is a canonical NFA for a given regular language? Answering this question is of great practical importance, as NFAs can be exponentially more succinct than their deterministic counterparts. In this case, the advantage is little, with the smallest DFA being of size 6, whereas a NFA can be constructed with 5 states. Generally, however, the difference increases with the size of the state-spaces. The problem has been approached independently, for various types of acceptors with side-effects. Most approaches have in common their restriction to a subclass that admits a canonical representative. Some of the better known examples are: the *átomaton* [39], the *canonical residual finite-state automaton* (short *canonical RFSA* and also known as *jiromaton*) [49], the *minimal xor automaton* [156], and the *distromaton* [119]. The canonical RFSA for the regular language $L = (a+b)^*a$ over the input alphabet $\{a, b\}$, for instance, is depicted in Figure 1.5b. It is minimal in the subclass of those NFAs accepting L, for which each state accepts a join of residuals of L. Its similarity to the minimal deterministic finite automaton for L in Figure 1.5a is striking.

Somewhat surprisingly, the state spaces of all the above canonical minimal representatives consist of *generators*, for different algebraic structures. For example, the

state-space of the átomaton is given by the *atoms*[2] for a complete atomic Boolean algebra, the states of the canonical RFSA are the *join-irreducibles*[3] of a complete semi-lattice (for example, the states in Figure 1.5b are the irreducibles of the lattice in Figure 4.6a), and the state-space of the minimal xor automaton consists of a *basis* for a vector-space. All these subsets of algebraic structures have in common that they contain the *minimal* amount of information to generate from it, by performing a closure with respect to algebraic operations, the full structure. For example, a subset of a vector space is called a basis for the former if every vector can be uniquely written as a finite linear combination of basis elements. Part of the importance of bases stems from the convenient consequences that follow from their existence. For instance, linear transformations between vector spaces admit matrix representations relative to pairs of bases, which can be used for efficient numerical calculations.

This observation immediately leads to numerous questions. Is the size-minimality of the generators underlying the state-spaces related to the size-minimality of the models? How are the underlying algebraic structures related to the type of side-effects of the models? What is the connection to the Myhill-Nerode construction for the minimal deterministic finite automaton? Is there a general procedure for the construction of minimal representatives?

Some of above questions were answered by Myers et al. [119], whose approach is based on an equivalence between finite algebras in a locally finite variety and finite structured sets and relations. Their construction, however, is restricted to non-deterministic automata, and does not provide a general algorithm to construct a succinct automaton. A different unifying perspective was given by van Heerdt [66, 63, 64]. One of the central notions in van Heerdt's work is the concept of a *scoop*, originally introduced by Arbib and Manes [19], and essentially a simple *category-theoretic* generalisation of algebraic generators. In this thesis, we refer to scoops as generators, and further develop their abstract theory.

[2] A non-zero element a in a Boolean algebra B is called an *atom*, if for all $x \in B$ with $x \leq a$ it follows $x = 0$ or $x = a$. A Boolean algebra B is *atomic*, if for all $x \in B$ there exists a decomposition $x = \vee_I a_i$, where $\{a_i \mid i \in I\}$ is some set of atoms.

[3] A non-zero element a in a lattice L is called *join-irreducible*, if for all $y, z \in L$ with $a = y \vee z$ it follows $a = y$ or $a = z$. For any x in a finite lattice L there exists a decomposition $x = \vee_I a_i$, where $\{a_i \mid i \in I\}$ is some set of join-irreducibles.

Set^T	TX
complete semi-lattices	$\{f : X \to 2\}$
$\mathbb{K}$-vector spaces	$\{f : X \to \mathbb{K} \mid \mathsf{supp}(f) \text{ is finite}\}$
complete atomic Boolean algebras	$\{f : (X \to 2) \to 2\}$
complete distributive lattices	$\{f : ((X \to 2), \subseteq) \to (2, \leq) \mid f \text{ is monotone}\}$

Figure 1.6: Algebraic structures as algebras over a monad on the category of sets

1.6 Categorical Perspective

Category theory is a mathematical framework that provides a unifying birds-eye perspective on different mathematical structures. The theory's central subjects are objects and the relations between them. Relations are treated as first class objects: they are *data*, rather than just a *property*. Examples of categories occur in all areas of mathematics and computer science. Some of the simplest cases are the categories sets and functions, and the category of vector spaces and linear maps. One of the attractive characteristics of category theory is that it allows simple unifying characterisations of constructions that deduce new mathematical objects from existing ones. For instance, one can show that the cartesian product is for sets and functions what the direct product is for groups and group homomorphisms. For this thesis, it will be sufficient to work with a relatively simple subset of category theory. We are particularly interested in *algebras* and *coalgebras*, and their combination into *bialgebras*.

Algebras In the category theoretic approach to universal algebra, algebraic structures are typically captured as *algebras* over a *monad* [52, 104].

In the context of computer science, monads have been introduced by Moggi, as a general perspective on exceptions, side-effects, and continuations [114, 113, 115]. Intuitively, they are a categorification of *closure operators*[4] on partially ordered sets. A simple example of a monad T on the category of sets is the *free* $\mathbb{K}$*-vector space monad* $\mathcal{V}_{\mathbb{K}}$, for any field $\mathbb{K}$. It assigns to a set X the set $\mathcal{V}_{\mathbb{K}}(X)$ of finitely-supported[5] func-

[4]Closure operators are monotone functions $T : P \to P$ that satisfy $x \leq T(x)$ and $T^2(x) = T(x)$ for all $x \in P$.

[5]The *support* of $\varphi : X \to \mathbb{K}$ is defined by $\mathsf{supp}(\varphi) = \{x \in X \mid \varphi(x) \neq 0\}$. If the set $\mathsf{supp}(f)$ is finite, then we say f has *finite support*, or is *finitely-supported*.

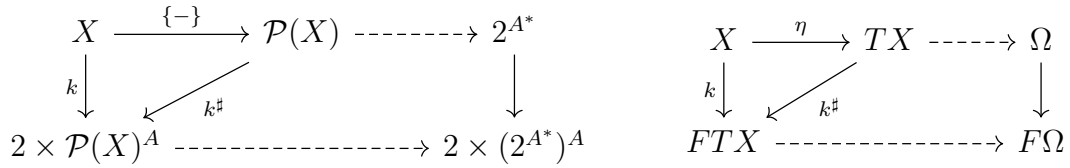


Figure 1.7: Generalised determinisation of automata with side-effects in a monad

tions $\varphi : X \to \mathbb{K}$; maps an element $x \in X$ to $\eta(x) \in \mathcal{V}_{\mathbb{K}}(X)$, the *Dirac measure*[6]; and flattens $\Phi \in \mathcal{V}^2_{\mathbb{K}}(X)$ to $\mu(\Phi) \in \mathcal{V}_{\mathbb{K}}(X)$ in the usual manner: if we write Φ as formal linear combination $\sum_\varphi \Phi_\varphi \cdot \varphi$, where Φ_φ is short for $\Phi(\varphi)$, then $\mu(\Phi)(x) = \sum_\varphi \Phi_\varphi \cdot \varphi(x)$. Most algebraic theories admit a monad on the category of sets by assigning to a set the set underlying the algebraic structure it freely generates. For example, the free vector space monad above is of this type.

An algebra over a monad T on the category of sets consists of a set X with a function $h : TX \to X$ that *interprets* elements in TX in a way that is coherent with the monad structure. A $\mathbb{K}$-vector space, for instance, is an algebra for the free $\mathbb{K}$-vector space monad $\mathcal{V}_{\mathbb{K}}$. It is given by a set X with a function $h : \mathcal{V}_{\mathbb{K}}(X) \to X$ that coherently interprets a finitely-supported function $\lambda : X \to \mathbb{K}$ (a *formal* linear combination) as an *actual* linear combination $h(\lambda) = \sum_x \lambda_x \cdot x \in X$ [44]. Any set X induces a *free* algebra TX over a set monad, by making use of the monad structure. A brief list of monads and the theories their algebras correspond to is given in Figure 1.6.

It is straightforward to see that under the above perspective a basis for a $\mathbb{K}$-vector space consists of a subset $Y \subseteq X$ and a function d that assigns to a vector $x \in X$ a finitely-supported function $d(x) \in \mathcal{V}_{\mathbb{K}}(Y)$ such that $h(d(x)) = x$ for all $x \in X$ and $d(h(\lambda)) = \lambda$ for all finitely-supported functions $\lambda : Y \to \mathbb{K}$. In other words, the restriction of h to finitely-supported functions with domain Y is a bijection with inverse d, and surjectivity corresponds to the fact that the subset Y generates the vector space, while injectivity captures that Y does so uniquely. The concept easily generalises to algebras over arbitrary monads on arbitrary categories by making the subset relation explicit. The generality also makes formal the intuitive observation that any set is a generator for the free algebra it induces.

[6]The Dirac measure $\eta(x) : X \to \mathbb{K}$ for $x \in X$ satisfies $\eta(x)(y) = 1$, if $x = y$, and 0 otherwise.

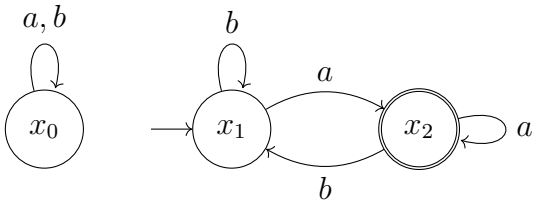
Coalgebras In the category theoretic approach to state-based systems, systems are typically captured as *coalgebras* over an *endofunctor* [78, 131, 132].

Endofunctors are categorifications of monotone functions $F : P \to P$ on a partially ordered set. They assign to any object X an object FX, and to every morphism $f : X \to Y$ a morphism $Ff : FX \to FY$. In particular, every monad thus is an endofunctor. A coalgebra over an endofunctor F on the category of sets consists of a set X with a function $k : X \to FX$. While algebras describe the *construction* of states, their dual notion, coalgebras, capture the *deconstruction* of states. A simple example of an endofunctor on the category of sets is the functor F that assigns to a set X the set $2 \times X^A$, where A is any fixed input set, and operates on functions as one expects. A coalgebra for it is simply an *unpointed* (i.e. without a specified initial state) deterministic automaton: the function k pairs the final state function and the transition function assigning a next state to each letter $a \in A$. In general, the definition of the functor F describes the *type*, or the *dynamics* of a system. By varying the underlying category and the type of F, one can recover many different types of transition systems.

There are many advantages to using the coalgebraic abstraction of state-based systems. Among others, it allows one to set aside irrelevant specifics of concrete instantiations, and instead work with elegant, universal properties. For instance, a central idea in the theory of systems is the notion of observable *behaviour*. In the coalgebraic formalism, the semantics of a system is conveniently captured as a unique structure-preserving function into a *final* coalgebra Ω. For example, for the above functor F with $FX = 2 \times X^A$, the final coalgebra is carried by the set of all languages $A^* \to 2$, and the final coalgebra homomorphism assigns to a state x of an unpointed deterministic automaton the language it accepts, when given the initial state x. This simplicity has lead to, among others, coalgebra successfully serving as a framework for the generalisation of automata learning algorithms in the style of L^* [80, 63, 64].

Bialgebras Of particular interest for us are systems that have both an algebraic and a coalgebraic component, interacting with each other in a well-defined way.

One simple example of such a system is the unpointed deterministic automaton $k^\sharp : \mathcal{P}(X) \to 2 \times \mathcal{P}(X)^A$ one obtains from determinising an unpointed non-deterministic



$\vee$	x_1	x_2	x_0
x_1	x_1	x_2	x_1
x_2	x_2	x_2	x_2
x_0	x_1	x_2	x_0

Figure 1.8: The minimal CSL-structured DFA accepting $(a+b)^*a \subseteq \{a,b\}^*$

automaton $k : X \to 2 \times \mathcal{P}(X)^A$ via the classical powerset-construction[7]. As one verifies, the coalgebraic structure of the lifting $k^\sharp$ has an additional property: it preserves the join semi-lattice structure $\cup : \mathcal{P}^2(X) \to \mathcal{P}(X)$ of its state-space. Indeed, a union $U \cup V$ is accepted if and only if either U or V is accepted; and reading in a at a state $U \cup V$ leads to a state $(U \cup V)_a$ that can equivalently be reached by taking the union $U_a \cup V_a$ of states reached by reading in a at U and V separately, respectively.

As seen on the right of Figure 1.7, the classical powerset-construction is an instance of a more general procedure which is parametric in an endofunctor F and a monad T, that interact via a *distributive law*[8] λ [140, 133]. Under this perspective, a *succinct* coalgebra $k : X \to FTX$ with side-effects in a monad T is transformed into a deterministic coalgebra $k^\sharp : TX \to FTX$ that interacts well with the freely generated algebra $\mu : T^2X \to TX$, and therefore is called a λ-*bialgebra*. An example of a bialgebra that intertwines the coalgebraic structure of a DFA with the algebraic structure of a complete semi-lattice (CSL) is depicted in Figure 1.8. Cases of succinct coalgebras are numerous, and include, aside of non-deterministic automata, probabilistic automata, nominal automata, and weighted automata. Active learning algorithms for succinct coalgebras have been studied by van Heerdt [67, 66, 66]. Distributive laws have originally been used to compose monads [25], but have since been generalised in a wide range of ways [147]. Bialgebras occur, among others, in a category-theoretic perspective on Structural Operational Semantics (SOS) [152, 86, 100, 75].

In this thesis, we study a construction that is reverse to the generalized determini-

[7]The powerset-construction assigns to an unpointed non-deterministic automaton $\langle \varepsilon, \delta \rangle : X \to 2 \times \mathcal{P}(X)^A$ the unpointed deterministic automaton $\langle \varepsilon^\sharp, \delta^\sharp \rangle : \mathcal{P}(X) \to 2 \times \mathcal{P}(X)^A$ defined by $\varepsilon^\sharp(U) = \vee_{u \in U} \varepsilon(u)$ and $\delta^\sharp(U)(a) = \cup_{u \in U} \delta(u)(a)$.

[8]A distributive law λ consists of a family of functions $\lambda_X : TFX \to FTX$, one for each set X, satisfying certain coherence conditions.

sation procedure. That is, we are looking for answers to the following questions. Can we transform a given bialgebra into a language equivalent succinct coalgebra that is size-minimal among all solutions? For instance, by exploiting the additional algebraic structure of the state-space of a bialgebra through an identification of algebraic generators? Is it possible to recover canonical automata from this transformation? Potentially, by identifying minimal generators for a minimal bialgebra?

1.7 Main Objectives

The objectives of this thesis are two-fold.

On the one hand, we would like to add to the automata (learning) theory of Guarded Kleene Algebra with Tests. It still being a relatively new logic, there are numerous central but open questions we plan to address, such as the existence of a minimal acceptor for a given language. Having the logic's potential applications in mind, a main goal of ours is to investigate variations and improve the efficiency of existing black-box learning algorithms that could be applied.

On the other hand, we aim to give a category-theoretic framework that unifies existing ways to construct minimal automata of different types. Canonical models of minimal size are the targets of automata learning: the design of algorithms often closely follows their construction. We expect this abstract approach to clarify and emphasise the currently under-appreciated role of algebraic generators, and lead to the discovery of new canonical acceptors. Part of our objective is to work generally enough such that our results are of value also outside the context of automata theory.

1.8 Overview and Contributions

This thesis consists of five parts. Chapter 1 and 2 contain an introduction and the preliminaries, respectively. The main contributions are given in chapter 3 to 5. Their content is as follows.

Chapter 3 In this chapter, we explore the automata theory of Guarded Kleene Algebra with Tests (GKAT). In particular, we present GL^* (Algorithm 2), an active

learning algorithm that derives a GKAT automaton representation of a black-box, by observing its behaviour through queries.

For any GKAT automaton, we define a second automaton, which we call its *minimisation* (Definition 3.4.2.1). In a series of results, we prove central properties about the minimisation of an automaton (Corollary 3.4.2.10, Lemma 3.4.2.7, Corollary 3.4.2.9, Corollary 3.4.2.8). We show that if GL^* is instantiated with the language accepted by a particular type of GKAT automaton, then the algorithm terminates with its minimisation in finite time (Theorem 3.5.2.6). We show that the semantics of GKAT automata can be reduced to the well-known semantics of Moore automata (Lemma 3.6.1.1, Corollary 3.6.1.2). A complexity analysis (Proposition 3.6.2.1) shows that it is more efficient to learn a representation with GL^* than with an existing variation of L^* for Moore automata. We implement GL^* and L^* in OCaml [122] and compare their performances on example programs (Figure 3.6).

The contributions of this chapter are based on the following publication (published at the 38th International Conference on Mathematical Foundations of Programming Semantics), of which the author of this thesis is the main author:

Stefan Zetzsche, Alexandra Silva, and Matteo Sammartino. "Guarded Kleene Algebra with Tests: Automata Learning". In: *Electronic Notes in Theoretical Informatics and Computer Science* Volume 1 - Proceedings of MFPS XXXVIII (2023). DOI: 10.46298/entics.10505

Chapter 4 In this chapter, we provide a general categorical framework based on bialgebras and distributive law homomorphisms that unifies constructions of canonical non-deterministic automata and unveils new ones.

We strictly improve the expressivity of previous work [67, 19] by including the átomaton (Section 4.5.3) and the distromaton (Section 4.5.4), which were previously excluded. While other frameworks restrict themselves to the category of sets [67], we are able to include canonical acceptors in other categories, such as the canonical *nominal* RFSA (Example 4.4.0.12). By relating vector spaces over the unique two element field with complete atomic Boolean algebras, we discover a previously unknown canonical mod-2 weighted acceptor for regular languages (Section 4.5.5). Finally, we show that every regular language satisfying a suitable property parametric in two

monads admits a size-minimal succinct acceptor (Theorem 4.6.0.5) and establish a size comparison between different acceptors (Section 4.6.1).

The contributions of this chapter are based on the following publication (published at the 37th International Conference on Mathematical Foundations of Programming Semantics), of which the author of this thesis is the main author:

Stefan Zetzsche, Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. "Canonical Automata via Distributive Law Homomorphisms". In: *Electronic Proceedings in Theoretical Computer Science* 351 (2021), 296–313. DOI: 10.4204/eptcs.351.18

Chapter 5 In this chapter, we show that the construction of canonical automata underlying our categorical framework is an instance of a more general theory.

First, we see (Example 5.2.5.2) that deriving a minimal bialgebra from a minimal coalgebra can be realized by applying a monad (Theorem 5.2.4.2) on a category of subobjects with respect to an epi-mono factorisation system (Definition 5.2.1.2). We then explore the abstract theory of generators and bases for algebras over a monad. We define a category of algebras with generators (Definition 5.3.1.1), which, we show, is in adjunction with the category of Eilenberg-Moore algebras (Lemma 5.3.1.2). We discuss products of generators and bases, and see that, under certain assumptions, the category of algebras with generators is monoidal (Corollary 5.3.2.2). The content of Section 5.3.3 is a generalisation of the representation theory of vector spaces. In Section 5.3.4 we discuss bases for bialgebras, which are algebras over a particular monad. A comparison of our ideas with an alternative approach that generalises bases as coalgebras is the content of Section 5.3.5. Signatures, equations, and finitary monads are discussed in Section 5.3.6. Finally, in Section 5.3.7, we relate our work to the theory of locally finitely presentable categories.

The contributions of this chapter are based on the following publication (published at the 10th Conference on Algebra and Coalgebra in Computer Science), of which the author of this thesis is the main author:

Stefan Zetzsche, Alexandra Silva, and Matteo Sammartino. "Generators and Bases for Monadic Closures". In: *arXiv preprint arXiv:2010.10223* (2023)

Chapter 2

Preliminaries

In this chapter, we will review the basic mathematical tools necessary to follow the more advanced constructions in this thesis. Readers familiar with the foundations of the theories of automata and categories may want to skip these pages.

2.1 Automata and Behaviour

The main subjects of this thesis are generalisations of automata and ways to construct them by observing the behaviour of a black-box system. In this section, we briefly recall the very basic definitions of (classical) automata theory. The presentation is entirely standard. For texts that present the full theory we recommend e.g. [94].

We begin with the definition of a non-deterministic automaton. Whether the non-deterministic or deterministic case is more elementary, thus should be given first, is a matter of taste, since the two notions are equivalent, as is well-known.

Definition 2.1.0.1 (Non-Deterministic Automaton)**.** A *non-deterministic automaton* (NA) is a tuple $\mathscr{X} = (X, \Sigma, \delta, \varepsilon, x_0)$ consisting of:

- a set X called *state space*;
- a finite non-empty set Σ called *alphabet*;
- a transition function $\delta : X \to \mathcal{P}(X)^{\Sigma}$;

- a function $\varepsilon : X \to 2$ characterising *accepting states*;
- an *initial state* $x_0 \in X$.

If the state space X is finite, we speak of a *non-deterministic finite automaton* (NFA).

A non-deterministic automaton for which the set $\delta(x)(a)$ consists of only one state, for all $x \in X$ and $a \in \Sigma$, is called a *deterministic automaton* (DA). A deterministic automaton with finite state space is called a *deterministic finite automaton* (DFA).

Every non-deterministic automaton $\mathscr{X}$ induces a deterministic automaton $\mathscr{X}^\sharp = (\mathcal{P}(X), \Sigma, \delta^\sharp, \varepsilon^\sharp, \{x_0\})$, via the *powerset construction*, defined by:

$$\delta^\sharp(U)(a) = \bigcup_{u \in U} \delta(u)(a), \qquad \varepsilon^\sharp(U) = \bigvee_{u \in U} \varepsilon(u).$$

Clearly $\mathscr{X}$ is finite if and only if $\mathscr{X}^\sharp$ is finite.

We say that a state $x \in X$ in a DA *transitions* to a state $y \in X$ via an input character $a \in \Sigma$, if $\delta(x)(a) = y$, in which case we write $x \xrightarrow{a} y$. A state $x \in X$ in a DA is *accepting*, if $\varepsilon(x) = 1$, and *rejecting* otherwise. The transition function $\delta : X \to X^\Sigma$ of a DA can be inductively extended to a function $\widehat{\delta} : X \to X^{\Sigma^*}$ that operates on words as follows:

$$\widehat{\delta}(x)(\varepsilon) = x, \qquad \widehat{\delta}(x)(av) = \widehat{\delta}(\delta(x)(a))(v).$$

For any $w \in \Sigma^*$, we write $\widehat{\delta}_w : X \to X$ for the function defined by $\widehat{\delta}_w(x) = \widehat{\delta}(x)(w)$. It is not hard to see that $\widehat{\delta}$ is a right-action of the free monoid Σ^* on the set X, that is, it satisfies $\widehat{\delta}_\varepsilon = \mathsf{id}_X$ and $\widehat{\delta}_{v \cdot w} = \widehat{\delta}_w \circ \widehat{\delta}_v$. A state $x \in X$ in a DA is *reachable* (from the initial state x_0), if there exists a word $w \in \Sigma^*$, such that $\widehat{\delta}(x_0)(w) = x$. A DA is *reachable*, if all of its states are reachable. Composing $\widehat{\delta}$ with the characterising function ε yields a function

$$[\![-]\!] = \varepsilon^{\Sigma^*} \circ \widehat{\delta} : X \to 2^{\Sigma^*}$$

that assigns to any state $x \in X$ its *behaviour*, or *semantics*, $[\![x]\!] \in 2^{\Sigma^*}$. A deterministic automaton is *observable*, if $[\![-]\!]$ is injective, that is, states can be distinguished by

observing their behaviour. A word $w \in \Sigma^*$ is *accepted* by a state $x \in X$ of a DA, if it induces a transition from x to an accepting state, $[\![x]\!](w) = 1$. The language accepted by a deterministic automaton $\mathscr{X}$ consists of the words accepted by its initial state,

$$[\![\mathscr{X}]\!] = [\![x_0]\!] : \Sigma^* \to 2.$$

Note that we sometimes implicitly identify characteristic functions with subsets, that is, in this case, we may speak of the *set* of accepted words $[\![\mathscr{X}]\!] \subseteq \Sigma^*$. We call two deterministic automata $\mathscr{X}$ and $\mathscr{Y}$ *language equivalent*, if they accept the same languages, $[\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$.

The semantics of a NA $\mathscr{X}$ is defined in terms of its determinisation, that is, $[\![\mathscr{X}]\!] = [\![\mathscr{X}^\sharp]\!]$. It is not hard to see that NFA and DFA accept the same class of languages: the *regular* languages.

Regular languages are unique in that they have a particularly nice property: for any regular language, there exists a (uniquely defined up-to isomorphism) deterministic finite automaton that accepts it and has the smallest state-space among all deterministic finite automata with the same behaviour. To this end, we make the following definition.

Definition 2.1.0.2 (Minimal Deterministic Automaton)**.** The *minimal* deterministic automaton for a language $L \subseteq \Sigma^*$ is the deterministic automaton $\mathsf{M}_L = (\mathsf{Der}(L), \delta, \varepsilon, L)$, where:

- $\mathsf{Der}(L) = \{w^{-1}L \mid w \in \Sigma^*\}$;
- $w^{-1}L = \{v \in \Sigma^* \mid wv \in L\}$;
- $\delta(w^{-1}L)(a) = (wa)^{-1}L$;
- $\varepsilon(w^{-1}L) = \begin{cases} 1 & w \in L \\ 0 & \text{else} \end{cases}$.

The states of the minimisation are called *derivatives* or *residuals* (of L) and correspond to the equivalence classes of the *Myhill-Nerode relation* $\simeq_L \subseteq \Sigma^* \times \Sigma^*$:

$$w_1 \simeq_L w_2 :\Leftrightarrow w_1^{-1}L = w_2^{-1}L.$$

A central result of classical automata theory says that the Myhill-Nerode relation $\simeq_L$ admits finitely many equivalence classes if and only if L is regular.

Lemma 2.1.0.3 ([120]). *M_L is finite if(f) L is regular.*

The minimal DFA M_L for a regular language L is reachable and observable, and it accepts L. It deserves its name since it is minimal in the following sense: For any DFA $\mathscr{X}$ accepting L, it holds $|\mathsf{M}_L| \leq |\mathscr{X}|$ with $|\mathsf{M}_L| = |\mathscr{X}|$ if and only if there is an isomorphism $\mathsf{M}_L \cong \mathscr{X}$. Note that this implies that all DFA of minimal-size accepting L are isomorphic to M_L. As we will see later, the existence of a uniquely defined size-minimal acceptor in this sense is somewhat special to DFA.

2.2 Categories

In this section we give a brief introduction to the foundations of category theory. Our presentation is standard and may be skipped by readers familiar with categories, functors, natural transformations, adjoints, and monads. The scope of the presentation is limited by the applications of category theory in this thesis. Some elementary notions such as limits and Kan extensions are omitted. There are many great introductory texts that go into more depth, for instance [105, 21, 99].

Objects and Morphisms The central subjects of category theory are objects and the relations between them. Most notably, relations are not just a *property*, but are treated as *data*, witnessed by morphisms.

Definition 2.2.0.1 (Category). A *category* $\mathscr{C}$ consists of the following data:

- A class of *objects* $A, B, C, D, ..., X, Y, Z$
- A class of *morphisms* or *arrows* $f, g, h, ...$
- A binary operation that assigns to each morphism f two objects $\mathsf{dom}(f)$, and $\mathsf{cod}(f)$ called the *domain* and *codomain* of f, respectively. The expression $f : X \to Y$ indicates that $X = \mathsf{dom}(f)$ and $Y = \mathsf{cod}(f)$.

- A binary operation that assigns to any two morphisms $f : X \to Y$ and $g : Y \to Z$ a morphism $g \circ f : X \to Z$ called the *composition* of f and g.
- For any object X, there is a morphism $1_X = \mathsf{id}_X : X \to X$ called the *identitiy morphism* of X.

The data is required to satisfy the following constraints:

- If $f : A \to B$, $g : B \to C$ and $h : C \to D$, then $h \circ (g \circ f) = (h \circ g) \circ f$.
- If $f : X \to Y$, then $f \circ 1_X = f = 1_Y \circ f$.

The class of morphisms with domain X and codomain Y is denoted by $\mathsf{Hom}_{\mathscr{C}}(X, Y)$ or $\mathscr{C}(X, Y)$. If convenient, we omit parentheses and write fg for $f \circ g$. The expressions $X \in \mathscr{C}$ and $f \in \mathscr{C}$ indicate that X is an element of the class of objects of $\mathscr{C}$ and f is an element of the class of morphisms of $\mathscr{C}$, respectively.

A category is *locally small*, if its class of morphisms $\mathscr{C}(X, Y)$ is a set for any choice of objects, and *small*, if it is locally small and its class of objects is a set.

The canonical example of a category is Set, which has sets as objects and functions as morphisms. Many other examples stem from algebraic theories. In this thesis, we are mainly interested in the following cases:

- The category $\mathbb{K}$-Vect has as objects vector spaces over a field $\mathbb{K}$ and as morphisms $\mathbb{K}$-linear maps.
- The category CSL has as objects complete join-semi lattices, and as morphisms functions that preserve all joins.
- The category CABA has as objects complete atomic Boolean algebras, and as morphisms Boolean algebra homomorphisms that preserve all meets and joins.
- The category CDL has as objects completely distributive lattices, and as morphisms functions that preserve all meets and all joins.
- The category $\mathbb{A}$-Nom has as objects finitely supported nominal[1] $\mathbb{A}$-sets, and as

[1]Let $\mathsf{Perm}(\mathbb{A})$ be the set of *permutations* on $\mathbb{A}$, i.e. the bijective functions $\pi : \mathbb{A} \to \mathbb{A}$. A nominal $\mathbb{A}$-set is a set X equipped with an action of the permutation group $(\mathsf{Perm}(\mathbb{A}), \mathsf{id}_{\mathbb{A}}, \circ)$. We say that a nominal set has *finite support*, if for each $x \in X$, there exists a finite set $A_x \subseteq \mathbb{A}$ such that for all $\pi \in \mathsf{Perm}(\mathbb{A})$ with $\pi.a = a$ for all $a \in A_x$, we have $\pi.x = x$. A function $f : X \to Y$ between nominal sets is *equivariant* if $f(\pi.x) = \pi.f(x)$ for all $\pi \in \mathsf{Perm}(\mathbb{A}), x \in X$.

morphisms equivariant functions, for any countable set $\mathbb{A}$.

In some sense, categories are generalisations of monoids. Indeed, any monoid M induces a category that consists of one object, $\star$, and a family of morphisms $(f_m : \star \to \star)_{m \in M}$, composed by $f_m f_n = f_{mn}$. The category induced by the trivial monoid is called the *final category* and is denoted by 1_{Cat}. It consists of one object $\star$ and one morphism, the identity $1_\star$.

Categories induce, and can be composed to, new categories. For example, the *opposite* or *dual* category $\mathscr{C}^{\mathrm{op}}$ of a category $\mathscr{C}$ has the same objects as $\mathscr{C}$, but an arrow $f : X \to Y$ in $\mathscr{C}^{\mathrm{op}}$ is an arrow $\tilde{f} : Y \to X$ in $\mathscr{C}$. Another example is the *product category* $\mathscr{C} \times \mathscr{D}$ of categories $\mathscr{C}, \mathscr{D}$, which consists of pairs of objects and pairs of morphisms, and component-wise defined composition.

Diagrammatical Proofs Many proofs in this thesis are given diagrammatically, via a method called *diagram chasing*. To this end, assume we are given the following objects and morphisms between them:

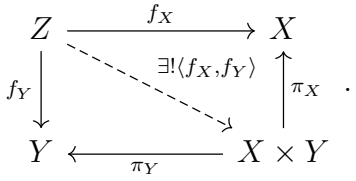
$$\begin{array}{ccccc} A & \xrightarrow{f} & B & \xrightarrow{g} & C \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle l} & & \downarrow{\scriptstyle h} \\ D & \xrightarrow[j]{} & E & \xrightarrow[k]{} & F \end{array}$$

We say that the outer diagram *commutes*, if $hgf = kji$. To prove that the outer diagram commutes, it is sufficient to show that the two inner diagrams commute, that is, $lf = ji$ and $hg = kl$. Diagrammatical proofs *divide and conquer*: they slice larger diagrams into smaller diagrams whose commutativity is known.

Universal Properties Many constructions in category theory are given in terms of *universal properties*, that define an object, if it exists, uniquely up to unique isomorphism. Below we give a few basic examples of such characterisations:

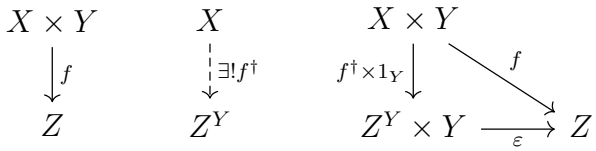
- A *product* of two objects X, Y consists of an object $X \times Y$ and two morphisms $\pi_X : X \times Y \to X$ and $\pi_Y : X \times Y \to Y$ that satisfy the following universal property: For every object Z and two morphisms $f_X : Z \to X$ and $f_Y :$

$Z \to Y$, there exists a unique morphism $\langle f_X, f_Y \rangle : Z \to X \times Y$, such that $f_X = \pi_X \circ \langle f_X, f_Y \rangle$ and $f_Y = \pi_Y \circ \langle f_X, f_Y \rangle$:



The morphisms π_X, π_Y are referred to as the *projections* of the product. In the category of sets and functions, all binary products exist: they are given by the cartesian product $X \times Y = \{(x, y) \mid x \in X,\ y \in Y\}$ with its usual projections.

- An *exponential* of objects Y, Z in a category with binary products consists of an object Z^Y and a morphism $\varepsilon : Z^Y \times Y \to Z$, called *evaluation*, such that for any object X and morphism $f : X \times Y \to Z$ there is a unique arrow $f^\dagger : X \to Z^Y$ such that $\varepsilon \circ (f^\dagger \times 1_Y) = f$:



A category that has all finite products and exponentials is called *cartesian closed*. The category of sets and functions is cartesian closed: an exponential X^Y consists of all functions $f : Y \to X$, and ε satisfies $\varepsilon(f)(y) = f(y)$.

- An object 0 is *initial*, if for any object X there is a unique morphism $!_X : 0 \to X$. In the category of sets and functions, there is an initial object, the empty-set $\emptyset$.

Duality The theory of categories has built in a notion of *duality*, which admits for any construction a second one, of symmetric importance. Many well-known mathematical entities can be shown to come in such duality pairs.

One simple example of such a pair is given by the cartesian product and the disjoint union of sets. Indeed, by defining the *coproduct* $X + Y$ in $\mathscr{C}$ as the product $X \times Y$ in $\mathscr{C}^{\mathrm{op}}$, one can show that in Set the coproduct of two sets is given by their

disjoint union with its obvious embeddings.

Another example follows from defining an object 1 in $\mathscr{C}$ as *final*, if it is initial in $\mathscr{C}^{\mathrm{op}}$. That is, for any object X, there exists a unique morphism $!_X : X \to 1$. In the category Set, any singleton set $\{\star\}$ is a final object.

Functors At the heart of category theory lies the idea that relationships between entities are equally as important as the entities themselves. That is, for any type of object, there should exist a corresponding type of morphism that preserves their structure. The natural type of structure-preserving morphism that corresponds to categories is called a *functor* and defined below.

Definition 2.2.0.2 (Functor)**.** A *functor* $F : \mathscr{C} \to \mathscr{D}$ between categories $\mathscr{C}$ and $\mathscr{D}$ consists of the following data:

- For each object X in $\mathscr{C}$ there is an object $F(X)$ in $\mathscr{D}$.
- For each morphism $f : X \to Y$ in $\mathscr{C}$ there is a morphism $F(f) : F(X) \to F(Y)$ in $\mathscr{D}$.

The data is subject to the following constraints:

- If X is an object in $\mathscr{C}$, then $F(1_X) = 1_{F(X)}$.
- If $f : X \to Y$ and $g : Y \to Z$ are morphisms in $\mathscr{C}$, then $F(g \circ f) = F(g) \circ F(f)$.

If convenient, we omit parentheses and write FX for $F(X)$, and Ff for $F(f)$.

For every category $\mathscr{C}$ there exists an *identity functor* $1_{\mathscr{C}} : \mathscr{C} \to \mathscr{C}$, which maps objects and morphisms to themselves. The *composition* of two functors $F : \mathscr{C} \to \mathscr{D}$ and $G : \mathscr{D} \to \mathscr{E}$ is the functor $G \circ F : \mathscr{C} \to \mathscr{E}$ defined by $G \circ F(X) = G(F(X))$ and $G \circ F(f) = G(F(f))$. There exists a category, denoted by Cat, that has as objects small[2] categories and as morphisms functors between them. Many familiar categories can be recovered as universal constructions within this category. For example, the product category $\mathscr{C} \times \mathscr{D}$ can be recognised as the product of $\mathscr{C}, \mathscr{D}$ in Cat.

[2]For the same reason one cannot have a set of all sets, one can not construct a category that contains *all* categories as objects. A standard way to deal with the issue is to allow only *small* categories as objects. There are other ways (e.g. by using the language of higher category-theory) to avoid running into paradoxes. We will not encounter Cat again, thus refrain from elaborating.

There are numerous examples of functors, in every corner of mathematics. Simple generic cases are given by the *constant*[3] functor $F_X : \mathscr{C} \to \mathscr{D}$ (for any $X \in \mathscr{D}$), the *diagonal*[4] functor $\Delta : \mathscr{C} \to \mathscr{C} \times \mathscr{C}$, and the *product*[5] functor $\Pi : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$ (for any category $\mathscr{C}$ with binary products). A few more concrete examples are:

- The *dual vector space* functor $(-)^* : (\mathbb{K}\text{-}\mathsf{Vect})^{\mathrm{op}} \to \mathbb{K}\text{-}\mathsf{Vect}$ is defined on vector spaces over a field $\mathbb{K}$ by $V^* = \mathbb{K}\text{-}\mathsf{Vect}(V, \mathbb{K})$, and on linear maps f by $f^*(g) = gf$.

- The *free vector space* functor $\mathcal{V}_{\mathbb{K}} : \mathsf{Set} \to \mathbb{K}\text{-}\mathsf{Vect}$ over a field $(\mathbb{K}, +, \cdot)$ is defined on objects by $\mathcal{V}_{\mathbb{K}}(X) = \{\varphi : X \to \mathbb{K} \mid \mathsf{supp}(\varphi) \text{ is finite}^5\}$, equipped with the vector space structure induced by $\mathbb{K}$, and on morphisms $f : X \to Y$ by $\mathcal{V}_{\mathbb{K}}(f)(\varphi)(y) = \sum_{x \in f^{-1}(y)} \varphi(x) \in \mathbb{K}$. The *forgetful* functor $U : \mathbb{K}\text{-}\mathsf{Vect} \to \mathsf{Set}$ maps a vector space to its set of states, and a linear map to itself, viewed as function.

- The *free complete join-semi lattice* functor $\mathcal{P} : \mathsf{Set} \to \mathsf{CSL}$ is defined on sets X by $\mathcal{P}X = 2^X$, equipped with the join-semi lattice structure induced by 2, and on morphisms $f : X \to Y$ by $2^f(\varphi)(y) = \vee_{x \in f^{-1}(y)} \varphi(x) \in 2$. Equivalently, if we identify subsets with their characteristic functions, $\mathcal{P}X = \{U \mid U \subseteq X\}$ and $\mathcal{P}f(U) = \{f(u) \mid u \in U\}$. As before, the *forgetful* functor $U : \mathsf{CSL} \to \mathsf{Set}$ maps objects and morphisms to their underlying sets and functions.

Natural Transformations Morphisms relate objects, functors relate categories, and *natural transformations* relate functors. (Historically the interest in these notions has in fact been inverse to what the progression suggests. That is, Mac Lane had an interest in natural transformations in the context of homology that pre-dates the formal introduction of a functor [105].)

Definition 2.2.0.3 (Natural Transformation)**.** A *natural transformation* $\eta : F \Rightarrow G$ between functors $F, G : \mathscr{C} \to \mathscr{D}$ on categories $\mathscr{C}, \mathscr{D}$ consists of a family of morphisms

[3] For any object $X \in \mathscr{D}$, the constant functor $F_X : \mathscr{C} \to \mathscr{D}$ satisfies $F_X(Y) = X$ and $F_X(f) = 1_X$.

[4] The diagonal functor $\Delta : \mathscr{C} \to \mathscr{C} \times \mathscr{C}$ is defined by $\Delta(X) = (X, X)$ and $\Delta(f) = (f, f)$.

[5] For any category $\mathscr{C}$ with binary products, the product functor $\Pi : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$ is defined on objects by $\Pi(X, Y) = X \times Y$, and on morphisms $f : X \to X'$ and $g : Y \to Y'$ by $\Pi(f, g) = \langle f\pi_X, g\pi_Y \rangle : X \times Y \to X' \times Y'$.

$(\eta_X : FX \to GX)_{X \in \mathscr{C}}$ in $\mathscr{D}$, subject to the following *naturality* constraint: If $f : X \to Y$ is a morphism in $\mathscr{C}$, then the following diagram commutes:

$$\begin{array}{ccc} FX & \xrightarrow{\eta_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\eta_Y]{} & GY \end{array}$$

The class of natural transformations between F and G is denoted by $\mathrm{Nat}(F, G)$. For any functor F there exists an *identity transformation* $1_F \in \mathrm{Nat}(F, F)$ that is defined by $(1_F)_X = 1_{FX}$. A *natural isomorphism* is a natural transformation for which the morphism η_X is an isomorphism for every object X.

The classical example of a natural transformation is $\eta : 1_{\mathbb{K}\text{-}\mathsf{Vect}} \Rightarrow (-)^{**}$, given component-wise as the linear map $\eta_V : V \to V^{**}$ defined by $\eta_V(x) = \mathsf{ev}_x : V^* \to \mathbb{K}$, where $\mathsf{ev}_x(f) = f(x)$. If V is finite-dimensional, the embedding η_V is an isomorphism, $V \cong V^{**}$. Since the definition of η_V does not require the choice of a basis for V, it is canonical or *natural*. (On the other hand, any isomorphism witnessing $V \cong V^*$ for finite-dimensional V requires the choice of a basis, thus is *not* natural.)

Natural transformations compose with functors. That is, for any transformation $\eta : F \Rightarrow G$ between functors $F, G : \mathscr{C} \to \mathscr{D}$, and functor $H : \mathscr{D} \to \mathscr{E}$, there is a natural transformation $H\eta : HF \Rightarrow HG$ defined component-wise by $(H\eta)_X = H\eta_X$, and for any functor $K : \mathscr{B} \to \mathscr{C}$, there is a natural transformation $\eta_K : FK \Rightarrow GK$ defined by $(\eta_K)_X = \eta_{KX}$. Natural transformations can also be composed with each other, both *vertically*[6] and *horizontally*[7]. Vertical and horizontal composition satisfy the exchange law $(\varepsilon'\eta') \star (\varepsilon\eta) = (\varepsilon' \star \varepsilon)(\eta' \star \eta)$.

If $\mathscr{C}$ is a small category and $\mathscr{D}$ is any category, one can form the *functor category* $\mathscr{D}^{\mathscr{C}}$. Its objects are functors $F, G : \mathscr{C} \to \mathscr{D}$, and morphisms are natural transformations $\eta : F \Rightarrow G$, composed vertically. A natural transformation is an isomorphism in a functor category if and only if it is a natural isomorphism.

---

[6] If $\eta : F \Rightarrow G$ and $\varepsilon : G \Rightarrow H$ are natural transformation between functors $F, G, H : \mathscr{C} \to \mathscr{D}$, then their vertical composition $\varepsilon\eta : F \Rightarrow H$ is defined component-wise by $(\varepsilon\eta)_X = \varepsilon_X \circ \eta_X$.

[7] If $\eta : F \Rightarrow G$ is a natural transformation between functors $F, G : \mathscr{C} \to \mathscr{D}$ and $\varepsilon : J \Rightarrow K$ is a natural transformation between functors $J, K : \mathscr{D} \to \mathscr{E}$, then their horizontal composition $\varepsilon \star \eta : JF \Rightarrow KG$ is defined as the composition $\varepsilon \star \eta := \varepsilon_G J\eta$.

Adjoints In many situations, two categories are not isomorphic, but still closely related to each other. Often this weaker form equivalence can be captured by a notion called *adjointness*. Below, we give three different definitions, which all can be shown to be equivalent[8] to each other. Having different formulations at hand will make it easier for us to deduce structure relevant for later purposes.

Definition 2.2.0.4 (Adjunction 1)**.** We call a functor $F : \mathscr{D} \to \mathscr{C}$ *left adjoint* to a functor $G : \mathscr{C} \to \mathscr{D}$, and write $F \dashv G$, if there exists a natural isomorphism

$$\phi_{X,Y} : \mathscr{C}(FY, X) \cong \mathscr{D}(Y, GX).$$

between functors of type $\mathscr{D}^{\mathrm{op}} \times \mathscr{C} \to \mathsf{Set}$.

Definition 2.2.0.5 (Adjunction 2)**.** We call a functor $G : \mathscr{C} \to \mathscr{D}$ a *right adjoint* functor, if for each object Y in $\mathscr{D}$ there exists an object FY in $\mathscr{C}$ and a morphism $\eta_Y : Y \to GFY$ such that for every object X in $\mathscr{C}$ and every morphism $g : Y \to GX$, there exists a unique morphism $g^\sharp : FY \to X$ making the following diagram commute:

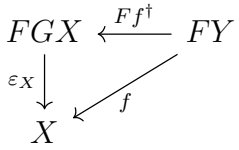
$$\begin{array}{ccc} GFY & \xrightarrow{Gg^\sharp} & GX \\ {\scriptstyle \eta_Y}\uparrow & \nearrow_{g} & \\ Y & & \end{array}$$

In the above situation, one can show that F extends to a functor $F : \mathscr{D} \to \mathscr{C}$ by defining it on morphisms $g : Y \to X$ as $Fg := (\eta_X \circ g)^\sharp$, and η extends to a natural transformation $\eta : 1_{\mathscr{D}} \Rightarrow GF$. The functor F is called a *left adjoint* to G.

Definition 2.2.0.6 (Adjunction 3)**.** We call a functor $F : \mathscr{D} \to \mathscr{C}$ a *left adjoint functor*, if for each object $X \in \mathscr{C}$ there exists an object GX in $\mathscr{D}$ and a morphism $\varepsilon_X : FGX \to X$ such that for every object Y in $\mathscr{D}$ and every morphism $f : FY \to X$,

[8]For instance, Definition 2.2.0.4 and Definition 2.2.0.5 are equivalent via the relations $\phi_{X,Y}(f) = G(f) \circ \eta_Y$ and $\eta_Y = \phi_{FY,Y}(1_{FY})$. Similarly, the equivalence of Definition 2.2.0.4 and Definition 2.2.0.6 follows from $\phi^{-1}_{X,Y}(g) = \varepsilon_X \circ F(g)$ and $\varepsilon_X = \phi^{-1}_{X,GX}(1_{GX})$.

there exists a unique morphism $f^\dagger : Y \to GX$ making the following diagram commute:



In the above situation, one can show that G extends to a functor $G : \mathscr{C} \to \mathscr{D}$ by defining it on morphisms $f : Y \to X$ as $Gf := (f \circ \varepsilon_Y)^\dagger$, and ε extends to a natural transformation $\varepsilon : FG \Rightarrow 1_{\mathscr{C}}$. The functor G is called a *right adjoint* to F.

The natural transformations η and ε are referred to as the *unit* and *counit* of the adjunction, respectively. We further define the natural transformations $\mu := G\varepsilon_F$ and $\delta := F\eta_G$. The unit η and the counit ε of an adjunction satisfy the *triangle identities* $(G\varepsilon)\eta_G = 1_G$ and $\varepsilon_F(F\eta) = 1_F$. Adjoints are unique up to isomorphism, that is, if $F \dashv G$ and $F \dashv H$, then there exists a natural isomorphism $G \cong H$. Adjunctions can be composed: if $F \dashv G$ is an adjunction between $\mathscr{C}$ and $\mathscr{D}$, and $J \dashv K$ an adjunction between $\mathscr{D}$ and $\mathscr{E}$, then one can show that $JF \dashv GK$ is an adjunction between $\mathscr{C}$ and $\mathscr{E}$. We conclude with a brief list of adjunctions between previously defined functors:

- Assume $\mathscr{C}$ has all binary products. Then $\Delta \dashv \Pi$, that is, the diagonal functor $\Delta : \mathscr{C} \to \mathscr{C} \times \mathscr{C}$ is left adjoint to the product functor $\Pi : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$.
- Assume $\mathscr{C}$ has all binary products and exponentials. Then $- \times Y \dashv (-)^Y$ for any object $Y \in \mathscr{C}$, i.e. the partial product functor $- \times Y : \mathscr{C} \to \mathscr{C}$ is left adjoint to the partial exponential $(-)^Y : \mathscr{C} \to \mathscr{C}$.
- The free vector space functor $\mathcal{V}_{\mathbb{K}} : \mathsf{Set} \to \mathbb{K}\text{-Vect}$ over a field $(\mathbb{K}, +, \cdot)$ is left adjoint to the forgetful functor $U : \mathbb{K}\text{-Vect} \to \mathsf{Set}$. The unit of the adjunction $\mathcal{V}_{\mathbb{K}} \dashv U$ is given by $\eta_X(x)(y) = [x = y]$ and μ satisfies $\mu_X(\Phi)(x) = \sum_{\varphi \in \mathcal{V}_{\mathbb{K}}(X)} \Phi(\varphi) \cdot \varphi(x)$ for $\Phi \in \mathcal{V}^2_{\mathbb{K}}(X)$

Monads Given an adjunction $F \dashv G$, it is not unnatural to ask what structure the endofunctors arising as the compositions GF and FG can be equipped with. Two particularly interesting choices are the structures (GF, η, μ) and $(FG, \varepsilon, \delta)$. Axiomatising the characteristics of the former choice leads to a structure called *monad*,

whereas the latter choice leads to a structure called *comonad*. While the two notions are dual, thus symmetric, we are particularly interested in the former case.

Definition 2.2.0.7 (Monad)**.** A *monad* on a category $\mathscr{C}$ is a tuple (T, η, μ) consisting of an endofunctor $T : \mathscr{C} \to \mathscr{C}$ and natural transformations $\eta : 1_{\mathscr{C}} \Rightarrow T$ and $\mu : T^2 \Rightarrow T$ satisfying the following two commutative diagrams:

$$\begin{array}{ccc} T^3 & \xrightarrow{\mu_T} & T^2 \\ {\scriptstyle T\mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ T^2 & \xrightarrow[\mu]{} & T \end{array} \qquad \begin{array}{ccc} T & \xrightarrow{\eta_T} & T^2 \\ {\scriptstyle T\eta}\downarrow & \searrow^{1_T} & \downarrow{\scriptstyle \mu} \\ T^2 & \xrightarrow[\mu]{} & T \end{array} . \tag{2.1}$$

By an abuse of notation we will refer to a monad by its underlying endofunctor.

A morphism $(F, \alpha) : (\mathscr{C}, S) \to (\mathscr{D}, T)$ between a monad S on a category $\mathscr{C}$ and a monad T on a category $\mathscr{D}$ consists of a functor $F : \mathscr{C} \to \mathscr{D}$ and a natural transformation $\alpha : TF \Rightarrow FS$ satisfying $\alpha \circ \eta^T F = F\eta^S$ and $F\mu^S \circ \alpha S \circ T\alpha = \alpha \circ \mu^T F$ [146]. The composition of monad morphisms $(F, \alpha) : (\mathscr{C}, S) \to (\mathscr{D}, T)$ and $(G, \beta) : (\mathscr{D}, T) \to (\mathscr{E}, U)$ is the monad morphism $(GF, G\alpha \circ \beta F) : (\mathscr{C}, S) \to (\mathscr{E}, U)$ [146].

A convenient way to view a monad on $\mathscr{C}$ is as categorified monoid in the category of endofunctors on $\mathscr{C}$, equipped with functor composition as monoidal product and the identity functor $1_{\mathscr{C}}$ as unit. Under this perspective, the constraints (2.1) are precisely the associativity and unit laws of a monoid, respectively.

We continue with a list of monads that will be relevant in the course of this thesis:

Examples 2.2.0.8.

- The *powerset* monad $\mathcal{P}$ on Set assigns to a set X the set $\mathcal{P}X = X \to 2$, and to a function f the function $\mathcal{P}f$ defined by $\mathcal{P}f(\varphi)(y) = \vee_{x \in f^{-1}(y)} \varphi(x)$. Its unit satisfies $\eta_X(x)(y) = [x = y]$, where $[x = y] = 1$, if $x = y$, and 0 otherwise, and its multiplication $\mu_X(\Phi)(x) = \vee_{\varphi \in 2^X} \Phi(\varphi) \wedge \varphi(x)$. Equivalently, if one identifies characteristic functions with subsets, $\mathcal{P}X = \{U \mid U \subseteq X\}$, $\mathcal{P}f(U) = \{f(u) \mid u \in U\}$, $\eta_X(x) = \{x\}$, and $\mu_X(\Phi) = \cup_{U \in \Phi} U$.

- The *free vector space* monad $\mathcal{V}_{\mathbb{K}}$ over a field $(\mathbb{K}, +, \cdot)$ on Set is defined by $\mathcal{V}_{\mathbb{K}}(X) = \{f : X \to \mathbb{K} \mid \mathsf{supp}(f) \text{ is finite}\}$ and $\mathcal{V}_{\mathbb{K}}(\varphi)(y) = \sum_{x \in f^{-1}(y)} \varphi(x)$. Its unit is given by $\eta_X(x)(y) = [x = y]$ and its multiplication by $\mu_X(\Phi)(x) = \sum_{\varphi \in \mathcal{V}_{\mathbb{K}}(X)} \Phi(\varphi) \cdot \varphi(x)$ for $\Phi \in \mathcal{V}^2_{\mathbb{K}}(X)$. The free vector space monad over the

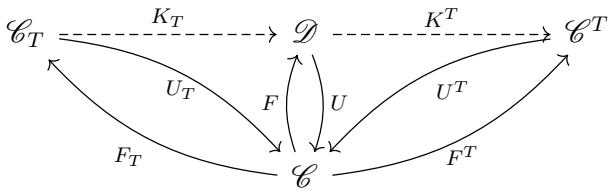


Figure 2.1: The category of adjunctions for a monad T on $\mathscr{C}$

unique two element field $(\mathbb{Z}_2, \oplus, \wedge)$ – with the *exclusive* disjunction $\oplus$ as addition, and the normal conjunction $\wedge$ as multiplication – is denoted by $\mathcal{R}$.

- The *neighbourhood* monad $\mathcal{H}$ on Set assigns to a set X the double dual function space $\mathcal{H}X = (X \to 2) \to 2$, and to a function f the function $\mathcal{H}f$ defined by $\mathcal{H}f(\Phi)(\varphi) = \Phi(\varphi \circ f)$. Its unit is given by $\eta^{\mathcal{H}}_X(x)(\varphi) = \varphi(x)$, and its multiplication satisfies $\mu^{\mathcal{H}}_X(\Psi)(\varphi) = \Psi(\eta^{\mathcal{H}}_{2^X}(\varphi))$.

- The *monotone neighbourhood* monad $\mathcal{A}$ on Set assigns to a set X the set of monotone functions $\mathcal{A}X = (X \to 2, \subseteq) \to (2, \leq)$, and otherwise coincides with the neighbourhood monad.

- The *nominal powerset* monad $\mathcal{P}_{\mathrm{n}}$ on $\mathbb{A}$-Nom assigns to a nominal set X the nominal set $\mathcal{P}_{\mathrm{n}}X = \{B \subseteq X \mid B \text{ finitely supported}\}$, where $\pi.B := \{\pi.b \mid b \in B\}$, and otherwise coincides with the classical powerset monad $\mathcal{P}$ [125].

Algebras Every adjunction yields a monad. As it turns out, the inverse holds too. That is, given a monad, there is always an adjunction that yields it. In fact, there often is more than one adjunction giving rise to a monad. Below we construct the extreme solutions, using the Eilenberg-Moore and Kleisli categories.

Definition 2.2.0.9 (Eilenberg-Moore Category)**.** An *algebra over a monad* T on $\mathscr{C}$ is a pair (X, h) consisting of an object X and a morphism $h : TX \to X$ in $\mathscr{C}$ satisfying the following two commutative diagrams:

$$\begin{array}{ccc} T^2X & \xrightarrow{Th} & TX \\ {\scriptstyle \mu_X}\downarrow & & \downarrow{\scriptstyle h} \\ TX & \xrightarrow[h]{} & X \end{array} \qquad \begin{array}{ccc} X & & \\ {\scriptstyle \eta_X}\downarrow & \searrow^{\mathsf{id}_X} & \\ TX & \xrightarrow[h]{} & X \end{array} .$$

A homomorphism $f : (X, h_X) \to (Y, h_Y)$ between T-algebras is a morphism $f : X \to Y$ such that $h_Y \circ Tf = f \circ h_X$. The *Eilenberg-Moore category* consists of T-algebras and T-algebra homomorphisms and is denoted by $\mathscr{C}^T$.

The categories $\mathscr{C}$ and $\mathscr{C}^T$ can be related as follows. On the one hand, there is the forgetful functor $U^T : \mathscr{C}^T \to \mathscr{C}$, defined by $U^T(X, h) = X$ and $U^T(f) = f$. On the other hand, there is the free algebra functor $F^T : \mathscr{C} \to \mathscr{C}^T$, which satisfies $F^T(X) = (TX, \mu_X)$ and $F^T(f) = Tf$. One can show that the pair satisfies an adjoint relation $F^T \dashv U^T$ that gives rise to T, and is final among all adjoint pairs inducing T. That is, for any adjunction $F \dashv U$ between functors with induced monad $T = UF$, there exists a comparison functor $K^T : \mathscr{D} \to \mathscr{C}^T$ that has the property $U^T K^T = U$ and $K^T F = F^T$. One can show that K^T is the unique functor with this property. An adjunction is *monadic*, if its comparison functor is an isomorphism.

The adjoint pair at the other end of the spectrum can be constructed as follows.

Definition 2.2.0.10 (Kleisli Category)**.** The *Kleisli category* for a monad T on $\mathscr{C}$, denoted by $\mathscr{C}_T$, has the same objects as $\mathscr{C}$; a morphism $f : X \nrightarrow Y$ in $\mathscr{C}_T$ is a morphism $f : X \to TY$ in $\mathscr{C}$; and the composition of $f : X \nrightarrow Y$ and $g : Y \nrightarrow Z$ in $\mathscr{C}_T$ is the composition $\mu_Z \circ Tg \circ f : X \to TZ$ in $\mathscr{C}$.

Analogously as before, the categories $\mathscr{C}$ and $\mathscr{C}_T$ can be related via two functors. On the one hand, there is the functor $U_T : \mathscr{C}_T \to \mathscr{C}$, defined by $U_T(X) = TX$ and $U_T(f : X \nrightarrow Y) = \mu_Y \circ Tf$. On the other hand, there is the functor $F_T : \mathscr{C} \to \mathscr{C}_T$, defined by $F_T(X) = X$ and $F_T(f : X \to Y) = \eta_Y \circ f$. The two functors satisfy an adjoint relation $F_T \dashv U_T$ that gives rise to the monad $T = U_T F_T$, and for any adjunction $F \dashv U$ with induced monad $T = UF$, there exists a comparison functor $K_T : \mathscr{C}_T \to \mathscr{D}$ that is uniquely defined by the property $UK_T = U_T$ and $K_T F_T = F$.

Figure 2.1 summarises the initiality of the Kleisli category $\mathscr{C}_T$ and the finality of the Eilenberg-Moore category $\mathscr{C}^T$ among adjoint pairs giving rise to T.

Often, the Eilenberg-Moore category of algebras over a monad can be recognised as a category of algebras over some theory in the sense of universal algebra. (This observation can be formalised, see e.g. Section 5.3.6.) Below, we list the identifications that are of particular interest for this thesis:

Example 2.2.0.11.

- The category $\mathsf{Set}^{\mathcal{P}}$ is isomorphic to CSL, the category of complete join-semi lattices and functions that preserve all joins (see e.g. [77]).
- The category $\mathsf{Set}^{\mathcal{H}}$ is isomorphic to CABA, the category of complete atomic Boolean algebras and Boolean algebra homomorphisms that preserve all meets and all joins (see e.g. [76]).
- The category $\mathsf{Set}^{\mathcal{A}}$ is isomorphic to CDL, the category of completely distributive lattices and functions that preserve all meets and all joins (see e.g. [76]).
- The category $\mathsf{Set}^{\mathcal{R}}$ is isomorphic to $\mathbb{Z}_2$-Vect, the category of vector spaces over the unique two element field and linear maps (see e.g. [77]).

Coalgebras We conclude with a few words about *coalgebras*. As the name suggests, coalgebras are dual to algebras. However, one typically refrains from requiring the dual of the Eilenberg-Moore laws, and instead simply works with coalgebras over an endofunctor (instead of a comonad). Over the last few years, the theory of coalgebras has become increasingly popular as a unifying framework for the study of infinite data types and state-based systems [132].

Definition 2.2.0.12 (Coalgebra)**.** A *coalgebra* for an endofunctor F on a category $\mathscr{C}$ is a pair (X, k) consisting of an object X and a morphism $k\colon X \to FX$ in $\mathscr{C}$.

If (X, k) is a F-coalgebra and $x \in X$, we call the tuple (X, k, x) either a *x-pointed F-coalgebra*, or an *F-automaton*.

One of the most basic examples of coalgebras in the category of sets and functions are unpointed deterministic automata: they are of the type $k\colon X \to FX$, where $FX = 2 \times X^A$ and k pairs the final state function and the transition function assigning a next state to each letter $a \in A$.

Crucial in the theory of coalgebras is the notion of homomorphism, which allows to relate states of coalgebras of the same behaviour. A homomorphism $f\colon (X, k_X) \to (Y, k_Y)$ between F-coalgebras is a morphism $f\colon X \to Y$ satisfying $k_Y \circ f = Ff \circ k_X$. The category of F-coalgebras and homomorphisms is denoted by $\mathsf{Coalg}(F)$.

If it exists, the final object of this category is of particular importance.

Definition 2.2.0.13 (Final Coalgebra)**.** An F-coalgebra (Ω, k_Ω) is *final* if every F-coalgebra (X, k) admits a unique homomorphism $\mathsf{obs}_{(X,k)} : (X, k) \to (\Omega, k_\Omega)$.

The unique final coalgebra homomorphism can be understood as the observable behaviour of a system. For example, for the functor $FX = 2 \times X^A$, the final F-coalgebra is the set of all languages $\mathcal{P}(A^\star)$ and the final coalgebra homomorphism assigns to a state x of an unpointed deterministic automaton the language in $\mathcal{P}(A^*)$ it accepts[9] when given the initial state x. More generally, for any F with $FX = B \times X^A$, the final F-coalgebra exists. Its underlying state-space is the set of generalised languages $A^* \to B$. We say that a x-pointed F-coalgebra (X, k, x) *accepts* the generalised language $\mathsf{obs}_{(X,k)}(x) \in B^{A^*}$.

In the course of this thesis we will encounter situations that require us to to deploy simultaneously both an algebraic and a coalgebraic perspective.

[9]For a deterministic automaton given by $\varepsilon : X \to 2$ and $\delta : X \to X^A$, acceptance is coinductively defined as a function $\mathsf{obs} : X \to 2^{A^*}$ by $\mathsf{obs}(x)(\varepsilon) = \varepsilon(x)$ and $\mathsf{obs}(x)(av) = \mathsf{obs}(\delta(x)(a))(v)$.

Chapter 3

Learning Guarded Programs

Guarded Kleene Algebra with Tests (GKAT) is the fragment of Kleene Algebra with Tests (KAT) that arises by replacing the union and iteration operations of KAT with predicate-guarded variants. GKAT is more efficiently decidable than KAT and expressive enough to model simple imperative programs, making it attractive for applications to e.g. network verification. In this chapter, we further explore GKAT's automata theory, and present GL^*, an algorithm for learning the GKAT automaton representation of a black-box, by observing its behaviour. A complexity analysis shows that it is more efficient to learn a representation of a GKAT program with GL^* than with Angluin's existing L^* algorithm. We implement GL^* and L^* in OCaml and compare their performances on example programs.

3.1 Introduction

As hardware and software systems grow in size and complexity, practical and scalable methods for verification tasks become increasingly important. Classical model checking approaches to verification require a rich model of the system of interest, able to express all its relevant behaviour. In reality such a model however is rarely available, for instance, when the system comes in the form of a black-box with no access to the source code, or the system is simply too complex for manual processing.

Automata learning, or regular inference, aims to automatically infer an automata model by observing the behaviour of the system. The incremental approach has

been successfully applied to a wide range of verification tasks from finding bugs in network protocols [21], reverse engineering smartcard reader for internet banking [42], and industrial applications [61]. A comprehensive survey of the field can be found in [153]. The majority of modern learning algorithms is based on Angluin's L^* algorithm [14], which learns the unique minimal deterministic finite automaton (DFA) accepting a given regular language, or more generally, the unique minimal Moore automaton accepting a weighted language (Algorithm 1). In many situations, however, targeting a DFA is not feasible, due to an explosion in the size of the state-space. Such cases instead require types of models specifically tailored for their domain-specific purposes.

For instance, modern networking systems can operate on very large data sets, making them very challenging to model. As a result, controlling, reasoning about, or extending networks can be surprisingly difficult. One approach to modernise the field that has recently gained popularity is *Software Defined Networking* (SDN) [55]. Modern SDN programming languages, notably *NetKAT* [12], allow operators to model their network and dynamically fine tune forwarding behaviour in response to events such as traffic shifts. Globally, NetKAT is based on *Kleene Algebra* (KA) [88], the sound and complete theory of regular expressions [84]. Locally, it incorporates *Boolean algebra*, the theory of predicates. Both logics have been unified in the well developed theory of *Kleene Algebra with Tests* (KAT) [90], which subsumes propositional Hoare logic and can be used to model standard imperative programming constructs. The automata theory for NetKAT has been introduced in [57].

Verifying properties about realistic networks reduces in NetKAT to deciding the behavioural equivalence of pairs of automata. Unfortunately, NetKAT's decision procedure is PSPACE-complete, mainly due its foundations in KAT. As a consequence, more efficiently decidable fragments of KAT have been considered. In [143] it was hinted that the *guarded fragment* of KAT is notably more efficiently decidable than the full language, while still remaining sufficiently expressive for networking purposes. The idea has been taken further in [142], which formally introduced *Guarded Kleene Algebra with Tests* (GKAT), a variation on KAT that arises by replacing the union and iteration operations from KAT with guarded variants. In contrast to KAT, the equational theory of GKAT is decidable in (almost) linear time. These properties make GKAT a promising candidate for the foundations of a SDN programming lan-

Algorithm 1 Angluin's L^* algorithm for Moore automata with input alphabet A and output alphabet B

$S, E \leftarrow \{\varepsilon\}$
repeat
  while $T = (S, E, \mathsf{row} : S \cup S \cdot A \to B^E)$ is not closed **do**
    find $t \in S \cdot A$ with $\mathsf{row}(t) \neq \mathsf{row}(s)$ for all $s \in S$
    $S \leftarrow S \cup \{t\}$
  end while
  construct and submit $m(T)$ to the teacher
  if the teacher replies *no* with a counterexample $z \in A^*$ **then**
    $E \leftarrow E \cup \mathsf{suf}(z)$
  end if
until the teacher replies *yes*
return $m(T)$

guage that is more efficiently decidable than NetKAT.

In view of the potential applications of GKAT to the field of verification, this chapter further investigates its automata theory. In detail, we make the following contributions:

- For any GKAT automaton, we define a second automaton, which we call its minimisation (Definition 3.4.2.1). We show that in the class of *normal* automata, the minimisation of an automaton is the unique size-minimal normal automaton accepting the same language (Corollary 3.4.2.10). We show that the minimisation of a normal automaton is isomorphic to the automaton that arises by identifying semantically equivalent pairs among reachable states (Lemma 3.4.2.7), and that the minimisations of two language equivalent normal automata are isomorphic (Corollary 3.4.2.9). Finally, we show that minimising a normal GKAT automaton preserves important invariants such as the nesting coequation (Corollary 3.4.2.8).

- We present GL^*, an active-learning algorithm (Algorithm 2) that incrementally infers a GKAT automaton from a black-box by querying an *oracle* (Section 3.5). We show that if the oracle is instantiated with the language accepted by a finite normal GKAT automaton, then the algorithm terminates with its minimisation in finite time (Theorem 3.5.2.6).

- We show that the semantics of GKAT automata (3.2) can be reduced to the well-known semantics[1] of Moore automata (3.6.1). That is, there exists a language preserving embedding of GKAT automata into Moore automata (Lemma 3.6.1.1), which maps the minimisation of a normal GKAT automaton to the language equivalent minimal Moore automaton (Corollary 3.6.1.2). In consequence, GKAT programs could thus, in principle, be also represented by Moore automata, instead of GKAT automata.

- We present a complexity analysis which shows that for GKAT programs it is more efficient to learn a GKAT automaton representation with GL^* than a Moore automaton representation with the L^* algorithm (Proposition 3.6.2.1). We implement GL^* and L^* in OCaml [122] and compare their performances on example programs (Figure 3.6).

3.2 Overview of the Approach

In this section, we give an overview of this chapter through examples. We begin by presenting Algorithm 1, a slight variation of Angluin's L^* algorithm for finite Moore automata. We exemplify the algorithm by executing it for the language semantics of a simple GKAT program. We then propose a new algorithm, which, instead of a Moore automaton, infers a GKAT automaton.

3.2.1 L^* Algorithm

Angluin's L^* algorithm learns the minimal DFA accepting a given regular language [14]. The algorithm has since been modified and generalised for a broad class of transition systems. The variation we present here step-wise infers the minimal Moore automaton accepting a generalised language $L : A^* \to B$ for an input alphabet A and

[1] In the language of Coalgebra, the semantics is given by the final coalgebra homomorphism for the functor defined by $FX = B \times X^A$, where $A = \mathsf{At} \cdot \Sigma = \{\alpha \cdot p \mid \alpha \in \mathsf{At},\ p \in \Sigma\}$ and $B = 2^{\mathsf{At}}$, for finite sets Σ and At. The carrier of the final coalgebra for F is $\mathcal{P}((\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At})$, the set of *guarded string languages*; the semantics of GKAT automata is given by the subclass of *deterministic* guarded string languages.

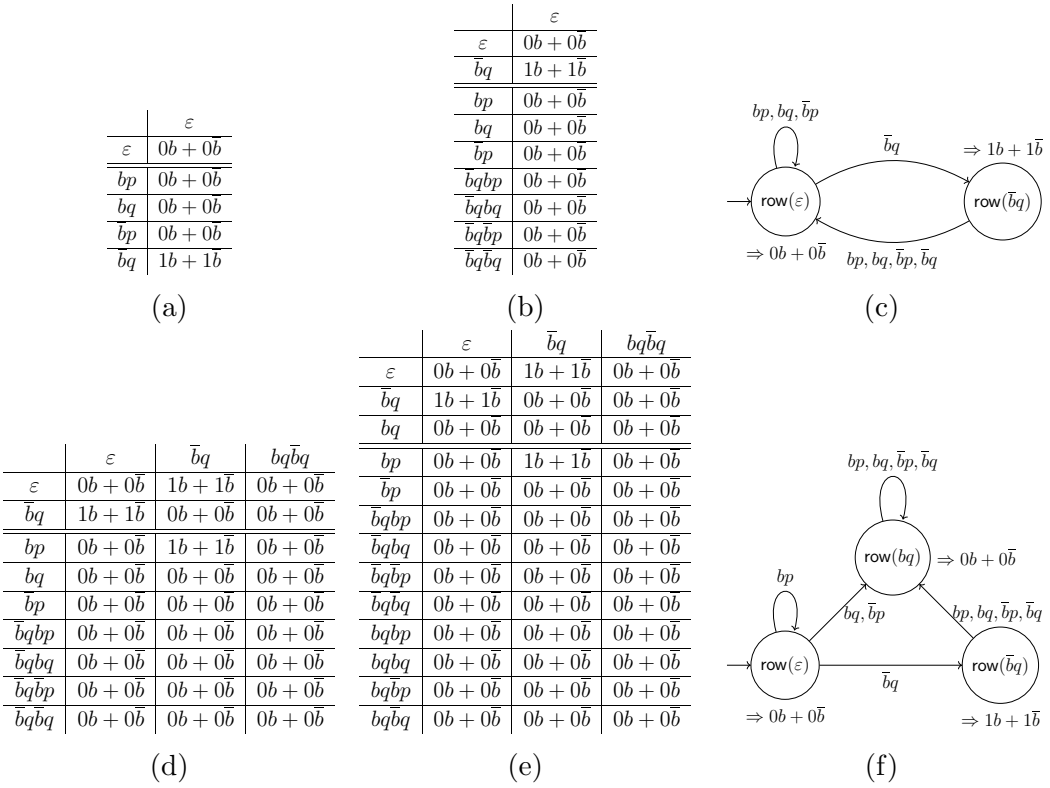


Figure 3.1: An example run of Angluin's L^* algorithm for the target language $[\![(\texttt{while}\ b\ \texttt{do}\ p); q]\!]$

an output alphabet B [118]. The algorithm assumes the existence of a *teacher* (or *oracle*), which can respond to two types of queries:

- **Membership queries**, consisting of a word $w \in A^*$, to which the teacher returns the output $L(w) \in B$;
- **Equivalence queries**, consisting of a hypothesis Moore automaton H, to which the teacher responds *yes*, if H accepts L, and *no* otherwise, providing a counterexample $z \in A^*$ in the symmetric difference of L and the behaviour of H.

The algorithm incrementally builds an *observation table*, which contains partial information about the language L obtained by performing membership queries. A table consists of two parts: a top part, with rows indexed by a finite set $S \subseteq A^*$; and a bottom-part, with rows ranging over $S \cdot A$. Columns are indexed by a finite set

$E \subseteq A^*$. For any $t \in S \cup S \cdot A$ and $e \in E$, the entry at row t and column e, denoted by $\mathsf{row}(t)(e)$, is given by the output $L(te) \in B$. Note that the sets S and $S \cdot A$ can intersect. In such a case, elements in the intersection are only shown in the top part. We refer to a table as a tuple $T = (S, E, \mathsf{row})$, leaving the language L implicit.

Given a table T, one can construct a Moore automaton $m(T) = (X, \delta, \varepsilon, x)$, where $X = \{\mathsf{row}(s) \mid s \in S\}$ is a finite set of states; the transition function $\delta : X \to X^A$ is given by $\delta(\mathsf{row}(s), a) = \mathsf{row}(sa)$; the output function $\varepsilon : X \to B$ satisfies $\varepsilon(\mathsf{row}(s)) = \mathsf{row}(s)(\varepsilon)$ (we abuse notation by writing ε both for the empty string and for the output function); and $x = \mathsf{row}(\varepsilon)$ is the initial state. For $m(T)$ to be well-defined, the table T has to satisfy $\varepsilon \in S$ and $\varepsilon \in E$, and two properties called closedness and consistency. An observation table is *closed* if for all $t \in S \cdot A$ there exists an $s \in S$ such that $\mathsf{row}(t) = \mathsf{row}(s)$. An observation table is *consistent*, if whenever $s, s' \in S$ satisfy $\mathsf{row}(s) = \mathsf{row}(s')$, then $\mathsf{row}(sa) = \mathsf{row}(s'a)$ for all $a \in A$. A table is consistent in particular if the function row is injective.

The algorithm incrementally updates the table to satisfy those properties. If a well-defined hypothesis $m(T)$ can be constructed, the algorithm poses an equivalence query to the teacher, and either terminates, or refines the hypothesis with a counterexample $z \in A^*$. Since we respond to a negative equivalence query by adding the suffixes[2] of a counterexample to the set E (opposed to adding the prefixes of a counterexample to the set S), rows will always be distinct, rendering consistency trivial[3]. At all times, the set S is prefix-closed and the set E is suffix-closed[4].

Example of execution

We now execute Angluin's L^* (Algorithm 1) for the target language

$$L = [\![(\texttt{while } b \texttt{ do } p); q]\!] = \{\overline{b}qb, \overline{b}q\overline{b}, bp\overline{b}qb, bp\overline{b}q\overline{b}, ...\} \subseteq (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}, \tag{3.1}$$

where $\mathsf{At} = \{b, \overline{b}\}$ is a finite set of *atoms* and $\Sigma = \{p, q\}$ is a finite set of *actions*. The language L represents the semantics of a program that performs the action p while b is true, and otherwise continues with q. It can be viewed as a generalised language

[2]The set $\mathsf{suf}(z)$ of suffixes for $z \in A^*$ is defined by $\mathsf{suf}(\varepsilon) = \{\varepsilon\}$ and $\mathsf{suf}(aw) = \{aw\} \cup \mathsf{suf}(w)$.
[3]This variation of L^* has been introduced by Maler and Pnueli [108].
[4]A set $X \subseteq A^*$ is called *suffix-closed*, if $\mathsf{suf}(z) \subseteq X$ for all $z \in X$.

$\widehat{L}$ with input alphabet $A = (\mathsf{At} \cdot \Sigma)$ and output alphabet $B = 2^{\mathsf{At}}$ via currying. We denote functions $f \in B$ as formal sums $\sum_{\alpha \in \mathsf{At}} f(\alpha)\alpha$. A query to $\widehat{L}$ requires $|\mathsf{At}|$ many queries to L.

Initially, the sets S and E are set to the singleton $\{\varepsilon\}$. We build the observation table in Figure 3.1a. Since the row indexed by $\overline{b}q$ does not appear in the upper part, i.e. differs from the row indexed by ε, the table is not closed. To resolve the closedness defect we add $\overline{b}q$ to S. The observation table (Figure 3.1b) is now closed. We derive from it the hypothesis depicted in Figure 3.1c. Next, we pose an equivalence query, to which the oracle replies *no* and informs us that the word $z = bq\overline{b}q$ has been falsely classified. Indeed, given z, the language accepted by the hypothesis outputs $1b + 1\overline{b}$, whereas (3.1) produces $0b + 0\overline{b}$. To respond to the counterexample z, we add its suffixes to E. In this case, there are only the two suffixes $\overline{b}q$ and $bq\overline{b}q$. The next observation table (Figure 3.1d) again is not closed: the row indexed by e.g. bq does not equal any of the two upper rows indexed by ε and $\overline{b}q$. To resolve the closedness defect we add bq to S, and obtain the table in Figure 3.1e. The observation table is now closed. We derive from it the automaton in Figure 3.1f. Next, we pose an equivalence query, to which the oracle replies *yes*.

3.2.2 GL^* Algorithm

In this section, we propose a new algorithm (Algorithm 2) for learning GKAT program representations, which we call GL^*. The new algorithm modifies Algorithm 1 by addressing a number of observations.

First, we note that the Moore automaton in Figure 3.1f admits multiple transitions to $\mathsf{row}(bq)$, a *sink-state*, which does not accept any words. Second, we observe that languages induced by GKAT programs are *deterministic*[5]. Such languages are naturally represented by GKAT automata, which keep some transitions implicit. Third, in some cases[6] the deterministic nature of the target language allows us to fill-in parts of the observation table without membership queries. Fourth, the cells of the obser-

[5]Deterministic in the sense that, whenever two strings agree on the first n atoms, then they agree on their first n actions (or lack thereof).

[6]For instance, the entries of the row indexed by bq in Figure 3.1d must all be zero, since the row indexed by bp admits a non-zero entry.

Algorithm 2 The GL^* algorithm for GKAT automata

$S \leftarrow \{\varepsilon\}, E \leftarrow \mathsf{At}$
repeat
  while $T = (S, E, \mathsf{row} : S \cup S \cdot (\mathsf{At} \cdot \Sigma) \to 2^E)$ is not closed **do**
    find $t \in S \cdot (\mathsf{At} \cdot \Sigma)$ with $\mathsf{row}(t)(e) = 1$ for some $e \in E$, but $\mathsf{row}(t) \neq \mathsf{row}(s)$ for all $s \in S$
    $S \leftarrow S \cup \{t\}$
  end while
  construct and submit $m(T)$ to the teacher
  if the teacher replies *no* with a counterexample $z \in (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$ **then**
    $E \leftarrow E \cup \mathsf{suf}(z)$
  end if
until the teacher replies *yes*
return $m(T)$

vation table are labelled by functions, each of which requires two membership queries to (3.1); as a consequence, table extensions require an unfeasible amount of queries.

As before, we assume two finite sets, At and Σ, and a deterministic language $L \subseteq (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$. The oracle of GL^* can answer two types of queries: membership queries consist of a word $w \in (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$, to which the oracle returns the output $L(w) \in 2$; equivalence queries consist of a hypothesis GKAT automaton H, to which the oracle responds *yes*, if H accepts L, and *no* otherwise, providing a counterexample $z \in (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$ in the symmetric difference of L and the language accepted by H.

An observation table in GL^* consists of two parts: a top part, with rows indexed by a finite set $S \subseteq (\mathsf{At} \cdot \Sigma)^*$; and a bottom-part, with rows ranging over $S \cdot \mathsf{At} \cdot \Sigma$. Columns range over a finite set $E \subseteq (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$. The entry of the observation table at row t and column e, denoted by $\mathsf{row}(t)(e)$, is given by $L(te) \in 2$. We refer to a table by $T = (S, E, \mathsf{row})$ and leave the deterministic language L implicit.

Given a table T, we construct an automaton $m(T) = (X, \delta, x)$, where $X = \{\mathsf{row}(s) \mid s \in S\}$ is a set of states; $x = \mathsf{row}(\varepsilon)$ is the initial state; and $\delta : X \to (2 + \Sigma \times X)^{\mathsf{At}}$ evaluates $\delta(\mathsf{row}(s))(\alpha)$ to $(p, \mathsf{row}(s\alpha p))$, if there exists an $e \in E$ and a $p \in \Sigma$ with $\mathsf{row}(s\alpha p)(e) = 1$; to 1, if $\mathsf{row}(s)(\alpha) = 1$; and to 0, otherwise.

Most of the properties a table needs to satisfy such that the hypothesis $m(T)$ is well-defined are guaranteed by the construction of Algorithm 2, since L is determin-

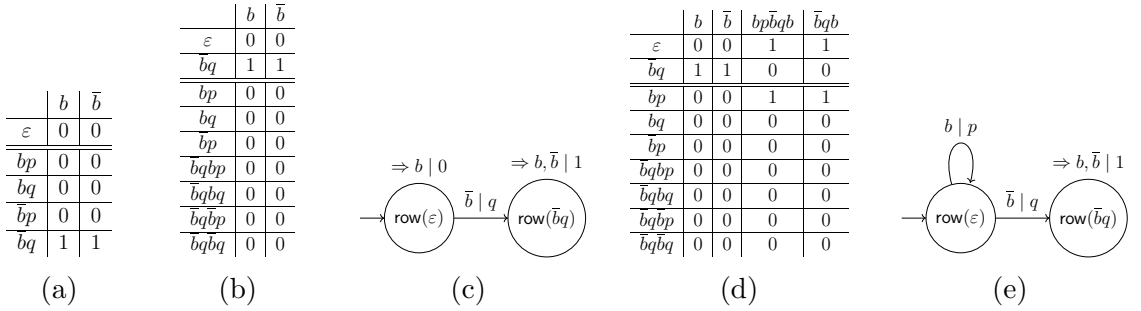


Figure 3.2: An example run of GL^* for the target language $[\![(\texttt{while}\ b\ \texttt{do}\ p); q]\!]$

istic. We only have to verify that the table is *closed*, that is, for all $t \in S \cdot \mathsf{At} \cdot \Sigma$ with $\mathsf{row}(t)(e) = 1$ for some $e \in E$, there exists some $s \in S$ such that $\mathsf{row}(t) = \mathsf{row}(s)$. As in the case of L^*, the algorithm incrementally updates the table until closedness is guaranteed. It then constructs a well-defined hypothesis, and poses an equivalence query to the teacher. If the oracle replies *yes*, the algorithm terminates, and if the response is *no*, it adds the suffixes[7] of a counterexample $z \in (\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$ to E.

The differences between GL^* and L^* (instantiated for $A = \mathsf{At} \cdot \Sigma$ and $B = 2^{\mathsf{At}}$) are essentially a consequence of currying. In the former case, the set E contains elements of type $(\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$, and the table is filled with booleans in 2; in the latter case, the set E contains elements of type $(\mathsf{At} \cdot \Sigma)^*$, and the table is filled with functions $\mathsf{At} \to 2$. This, however, does not mean that GL^* is merely a shift in perspective: its new types induce independent definitions, and termination needs to be established with novel correctness proofs (Section 3.5). A thorough comparison is given in Section 3.6.

Example of execution

We now execute Algorithm 2 for the target language (3.1). Initially, $S = \{\varepsilon\}$ and $E = \mathsf{At}$. We build the observation table in Figure 3.2a. Since the bottom row indexed by $\overline{b}q$ contains a non-zero entry and differs from all upper rows (in this case, only the row indexed by ε), the table is not closed. We resolve the closedness defect by adding $\overline{b}q$ to S. The observation table (Figure 3.2b) is now closed. Note that the row indexed by $\overline{b}q$ indicates that the words $\overline{b}qb$ and $\overline{b}q\overline{b}$ are accepted. Since we know the target language is deterministic, the last four rows of the table can be

[7]The set $\mathsf{suf}(z)$ of suffixes for $z \in A^* \cdot B$ is defined by $\mathsf{suf}(wb) = \{vb \mid v \in \mathsf{suf}(w)\}$.

filled with zeroes, without performing any membership queries. From Figure 3.2b we derive the hypothesis depicted in Figure 3.2c. Next, we pose an equivalence query, to which the oracle replies *no* and provides us with the counterexample $z = bp\overline{b}qb$, which is in the language (3.1), but not accepted by the hypothesis. We respond to the counterexample by adding its suffixes $bp\overline{b}qb$, $\overline{b}qb$ and b to E. The resulting observation table is depicted in Figure 3.2d. The table is closed, since the only non-zero bottom row is the one indexed by bp, which coincides with the upper row indexed by ε. Since the row indexed by bp has a non-zero entry, the row indexed by bq can automatically be filled with zeroes. We derive from Figure 3.2d the automaton in Figure 3.2e. Finally, we pose an equivalence query, to which the oracle replies *yes*.

3.3 Guarded Kleene Algebra with Tests

This section recalls the syntax and semantics of Guarded Kleene Algebra with Tests (GKAT). For most parts, we follow the relevant bits of the original presentation in [142]. We additionally introduce a notion of similarity between GKAT automata.

3.3.1 Syntax

The syntax of GKAT is inductively built from disjoint non-empty sets of *primitive tests*, T, and *actions*, Σ. In a first step, one generates from T a set of Boolean expressions, BExpr. In a second step, the set is extended with Σ, to the full set of GKAT expressions, Expr:

$$b, c, d \in \mathsf{BExpr} ::= 0 \mid 1 \mid t \in T \mid b \cdot c \mid b + c \mid \overline{b}$$
$$e, f, g \in \mathsf{Expr} ::= p \in \Sigma \mid b \in \mathsf{BExpr} \mid e \cdot f \mid e +_b f \mid e^{(b)}$$

By a slight abuse of notation, we will sometimes write ef for $e \cdot f$ and keep parenthesis implicit, e.g. $bc + d$ should be read as $(b \cdot c) + d$.

It is natural to view GKAT expressions as uninterpreted imperative programs. Under this view, one makes the identifications depicted in Figure 3.3.

Readers familiar with KAT will notice that the grammar for GKAT is similar to

$$0 \equiv \texttt{false} \qquad 1 \equiv \texttt{true} \qquad t \equiv t \qquad b \cdot c \equiv b \texttt{ and } c \qquad b + c \equiv b \texttt{ or } c$$
$$\overline{b} \equiv \texttt{not } b \qquad p \equiv \texttt{do } p \qquad b \equiv \texttt{assert } b \qquad e \cdot f \equiv e; f \qquad e^{(b)} \equiv \texttt{while } b \texttt{ do } e$$
$$e +_b f \equiv \texttt{if } b \texttt{ then } e \texttt{ else } f$$

Figure 3.3: Identifying GKAT expressions with imperative programs

the one of KAT. It differs in that GKAT replaces KAT's union ($+$) with the guarded union ($+_b$), and KAT's iteration (e^*) with the guarded iteration ($e^{(b)}$). GKAT's expressions can be encoded within KAT's grammar via the standard embedding that maps a conditional $e +_b f$ to $be + \overline{b}f$, and a while-loop $e^{(b)}$ to $(be)^*\overline{b}$.

3.3.2 Semantics: Language Model

In this section, we recall the language semantics of GKAT, which assigns to a program the traces it could produce once executed. Intuitively, an execution trace is a string of the shape $\alpha_0 p_1 \alpha_1 ... p_n \alpha_n$. It can be thought of as a sequence of states α_i a system is in at point i in time, beginning with α_0 and ending in α_n, intertwined with actions p_i that transition from the state α_{i-1} to the state α_i.

More formally, let $\equiv_{\mathsf{BA}}$ denote the equivalence relation between Boolean expressions induced by the Boolean algebra axioms. The quotient $\mathsf{BExpr}/_{\equiv_{\mathsf{BA}}}$, that is, the free Boolean algebra on generators T, admits a natural preorder $\leq$ defined by $b \leq c \Leftrightarrow b + c \equiv_{\mathsf{BA}} c$. The minimal nonzero elements with respect to this order are called *atoms*, the set of which is denoted by At. If $T = \{t_1, ..., t_n\}$ is finite, an atom $\alpha \in \mathsf{At}$ is of the form $\alpha = c_1 \cdot ... \cdot c_n$ with $c_i \in \{t_i, \overline{t_i}\}$.

A *guarded string* is an element of the set $\mathsf{GS} := \mathsf{At} \cdot (\Sigma \cdot \mathsf{At})^*$, or equivalently, $(\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}$. The set of guarded strings without terminating atom is $\mathsf{GS}^- := (\mathsf{At} \cdot \Sigma)^*$.

A guarded string language $L \subseteq \mathsf{GS}$ is *deterministic* [142, Def. 2.2], if, whenever $\alpha_1 p_1 ... \alpha_{n-1} p_{n-1} \alpha_n v \in L$ and $\alpha_1 q_1 ... \alpha_{n-1} q_{n-1} \alpha_n w \in L$, then $p_i = q_i$ for all $1 \leq i \leq n-1$, and either $v = w = \varepsilon$, or $v = p_n v'$ and $w = q_n w'$ with $p_n = q_n$. The set of deterministic guarded string languages is denoted by $\mathscr{L}$.

Guarded strings can be partially composed via the *fusion product* defined by $v\alpha \diamond \beta w := v\alpha w$, if $\alpha = \beta$, and undefined otherwise. The partial product lifts to a

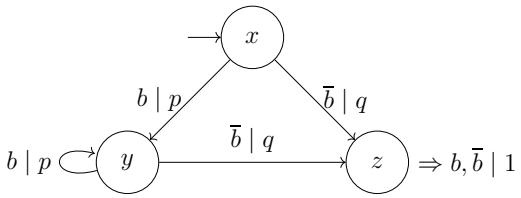


Figure 3.4: The Thompson-automaton $\mathscr{X}_{p^{(b)}q}$ for $T = \{b\}$ and $\Sigma = \{p, q\}$

total function on guarded languages by $L \diamond K := \{v \diamond w \mid v \in L, w \in K\}$. The n-th power of a guarded language is inductively defined by $L^0 := \mathsf{At}$ and $L^{n+1} := L^n \diamond L$. For $B \subseteq \mathsf{At}$ and $\overline{B} := \mathsf{At} \setminus B$, the guarded sum and the guarded iteration of languages are given by

$$L +_B K := (B \diamond L) \cup (\overline{B} \diamond K) \qquad L^{(B)} := \cup_{n \geq 0} (B \diamond L)^n \diamond \overline{B}.$$

The *language model* of GKAT is given by the semantic function $[\![-]\!] : \mathsf{Expr} \to \mathscr{P}(\mathsf{GS})$, which is inductively defined as follows:

$$[\![p]\!] := \{\alpha p \beta \mid \alpha, \beta \in \mathsf{At}\} \qquad [\![b]\!] := \{\alpha \in \mathsf{At} \mid \alpha \leq b\}$$
$$[\![e \cdot f]\!] := [\![e]\!] \diamond [\![f]\!] \qquad [\![e +_b f]\!] := [\![e]\!] +_{[\![b]\!]} [\![f]\!] \qquad [\![e^{(b)}]\!] := [\![e]\!]^{([\![b]\!])}.$$

Equivalently, the language semantics of GKAT can be constructed by post-composing the embedding of GKAT expressions into KAT expressions with the semantics of KAT. The language $[\![e]\!]$ accepted by a GKAT program e is deterministic.

Example 3.3.2.1. Let tests and actions be defined by $T := \{b\}$ and $\Sigma := \{p, q\}$, respectively. Then there exist only two atoms, $\mathsf{At} = \{b, \overline{b}\}$. The language model assigns to $p^{(b)}q \equiv (\texttt{while}\ b\ \texttt{do}\ p); q$ the guarded deterministic language in (3.1).

3.3.3 Semantics: Automata Model

In this section, we recall the automata model of GKAT, the central subject of this chapter. As before, we assume two finite sets of tests T and actions Σ, the former of which induces a finite set of atoms, At.

Let G be the functor on the category of sets which is defined on objects by $GX =$

$(2 + \Sigma \times X)^{\mathsf{At}}$ (where $2 = \{0, 1\}$) and on morphisms in the usual way. A *G-coalgebra* (cf. Definition 2.2.0.12) consists of a pair $\mathscr{X} = (X, \delta)$, where X is a set called *state-space* and $\delta : X \to GX$ is a function called *transition map*. A G-coalgebra *homomorphism* $f : (X, \delta^X) \to (Y, \delta^Y)$ is a function $f : X \to Y$ that commutes with the transition maps, $\delta^Y \circ f = Gf \circ \delta^X$. More concretely [142, Def. 5.7.], f is a G-coalgebra homomorphism, if for all $\alpha \in \mathsf{At}, p \in \Sigma$, and $x, y \in X$,

- if $\delta^X(x)(\alpha) \in 2$, then $\delta^Y(f(x))(\alpha) = \delta^X(x)(\alpha)$; and
- if $\delta^X(x)(\alpha) = (p, y)$, then $\delta^Y(f(x))(\alpha) = (p, f(y))$.

A *G-automaton* is a G-coalgebra $\mathscr{X}$ with a designated initial state $x \in X$. A homomorphism $f : (\mathscr{X}, x) \to (\mathscr{Y}, y)$ between G-automata is a homomorphism between the underlying G-coalgebras that maps initial state to initial state, $f(x) = y$.

For each state $x \in X$, given an input $\alpha \in \mathsf{At}$, a G-coalgebra either i) halts and accepts, that is, satisfies $\delta(x)(\alpha) = 1$; ii) halts and rejects, that is, satisfies $\delta(x)(\alpha) = 0$; or iii) produces an output p and moves to a new state y, that is, satisfies $\delta(x)(\alpha) = (p, y)$. Intuitively, for each state $x \in X$, a guarded string $\alpha_0 p_1 \alpha_1 ... p_n \alpha_n$ is accepted, if the G-coalgebra in state x produces the output $p_1 ... p_n$, halts and accepts. Formally, one defines a function $[\![-]\!] : X \to \mathscr{P}(\mathsf{GS})$ as follows:

$$\begin{aligned} \alpha \in [\![x]\!] &:\Leftrightarrow \delta(x)(\alpha) = 1; \\ \alpha p w \in [\![x]\!] &:\Leftrightarrow \exists y \in X : \delta(x)(\alpha) = (p, y) \text{ and } w \in [\![y]\!]. \end{aligned} \tag{3.2}$$

A G-coalgebra is *observable*, if the function $[\![-]\!]$ is injective.

A guarded string w is *accepted* by x, if $w \in [\![x]\!]$. The language accepted by a G-automaton, $[\![\mathscr{X}]\!]$, is the language accepted by its initial state $[\![x]\!]$. Every language accepted by a G-automaton is deterministic [142, Thm. 5.8]. Conversely, one can equip the set of deterministic languages with a G-coalgebra structure $(\mathscr{L}, \delta^{\mathscr{L}})$ defined by:

$$\delta^{\mathscr{L}}(L)(\alpha) = \begin{cases} (p, (\alpha p)^{-1} L) & \text{if } (\alpha p)^{-1} L \neq \emptyset \\ 1 & \text{if } \alpha \in L \\ 0 & \text{otherwise} \end{cases},$$

where $(\alpha p)^{-1}L = \{w \in \mathsf{GS} \mid \alpha pw \in L\}$. Note that $\delta^{\mathscr{L}}(L)$ is well-defined because L is deterministic. Since $[\![L]\!] = L$ for any $L \in \mathscr{L}$ [142, Thm. 5.8], every deterministic language can be recognized by a G-automaton with possibly infinitely many states.

A G-coalgebra (X, δ) is *normal*, if it only transitions to *live* states, that is, $\delta(x)(\alpha) = (p, y)$ implies $[\![y]\!] \neq \emptyset$. For any G-automaton $\mathscr{X}$ one can construct a language equivalent normal G-automaton $\widehat{\mathscr{X}}$ [142, Lem. 5.6]. If $\mathscr{X}$ is normal, the function $[\![-]\!] : X \to \mathscr{P}(\mathsf{GS})$ is the unique coalgebra homomorphism $[\![-]\!] : (X, \delta) \to (\mathscr{L}, \delta^{\mathscr{L}})$ [142, Thm. 5.8].

Two states $x, y \in X$ of a normal coalgebra accept the same language, $[\![x]\!] = [\![y]\!]$, if and only if they are *bisimilar*[8], $x \simeq y$, that is, there exists a binary relation $R \subseteq X \times X$ with xRy, such that if $x'Ry'$, then the following two implications hold:

- if $\delta(x')(\alpha) \in 2$, then $\delta(y')(\alpha) = \delta(x')(\alpha)$; and
- if $\delta(x')(\alpha) = (p, x'')$, then $\delta(y')(\alpha) = (p, y'')$ and $x''Ry''$ for some $y'' \in X$.

Bisimilarity is a symmetric relation and can be extended to two coalgebras by constructing a coalgebra that has the disjoint union of their state-spaces as state-space.

Using a construction that is reminiscent of Thompson's construction for regular expressions [150], it is possible to interpret a GKAT expression e as an automaton $\mathscr{X}_e$ that accepts the same language [142]. Alternatively, one can use a construction [142] that mirrors Kozen's syntactic form of Brzozowski's derivatives for KAT [91].

Example 3.3.3.1. The Thompson-automaton assigned to the expression $p^{(b)}q$ is depicted in Figure 3.4. It is normal and reachable, but not observable, since the states x and y are bisimilar, $x \simeq y$, thus accept the same language, $[\![x]\!] = [\![y]\!]$. It also is equivalent to the expression by which it is generated, that is, it satisfies $[\![\mathscr{X}_{p^{(b)}q}]\!] = [\![p^{(b)}q]\!]$.

3.3.4 A Note on Similarity

In this section we briefly introduce a notion of similarity that is to bisimulation, what a partial order is to equality. Our construction addresses the coalgebraic side of the proposal to replace the primitive notion of equality (equivalence) of GKAT

[8] In Section 3.3.4 we introduce the notion of similarity, which is to bisimilarity what a partial order is to equality.

expressions with a partial order of GKAT expressions [142]. We acknowledge that similarity has been studied more generally, for arbitrary coalgebras [24, 79, 69, 101].

Definition 3.3.4.1. Let $\mathscr{X}$ be a G-coalgebra. A *simulation* is a binary relation $R \subseteq X \times X$, such that if xRy, then:

- if $\delta(x)(\alpha) = 1$, then $\delta(y)(\alpha) = 1$;
- if $\delta(x)(\alpha) = (p, x')$, then $\delta(y)(\alpha) = (p, y')$ and $x'Ry'$ for some $y' \in X$.

States x and y are *similar*, $x \precsim y$, if there exists a simulation relating x to y.

The result below shows that similarity is more fundamental than bisimilarity, and that the definition of the latter naturally arises from the former.

Lemma 3.3.4.2. *$x \simeq y$ if and only if $x \precsim y$ and $y \precsim x$.*

Proof.

- Assume the bisimilarity $x \simeq y$ is witnessed by some relation R. We show that R witnesses the similarity $x \precsim y$. Clearly xRy by definition. Let $x'Ry'$ for arbitrary $x', y' \in X$, then we find:
  - If $\delta(x')(\alpha) = 1$, then $\delta(y') = 1$ since R is a bisimulation.
  - If $\delta(x')(\alpha) = (p, x'')$, then $\delta(y')(\alpha) = (p, y'')$ and $x''Ry''$ for some $y'' \in X$, since R is a bisimulation.

  Similarly, we show that the reverse relation R^r witnesses the similarity $y \precsim x$. Clearly yR^rx, since by construction xRy. Let $y'R^rx'$, i.e. $x'Ry'$, for arbitrary $x', y' \in X$, then we find:
  - If $\delta(y')(\alpha) = 1$, then $\delta(x')(\alpha) = 1$, since as R is a bisimulation we could otherwise falsely deduce $\delta(y')(\alpha) = 0$ or $\delta(y')(\alpha) \notin 2$.
  - If $\delta(y')(\alpha) = (p, y'')$, then $\delta(x')(\alpha) = (p, x'')$ with $y''R^rx''$, i.e. $x''Ry''$. Indeed, since R is a bisimulation, if $\delta(x')(\alpha) \in 2$, it falsely follows $\delta(y')(\alpha) \in 2$, and if $\delta(x')(\alpha) = (q, x'')$, it follows $\delta(y')(\alpha) = (q, y''')$ with $x''Ry'''$, as R is a bisimulation. It remains to observe $(p, y'') = \delta(y')(\alpha) = (q, y''')$, which implies $p = q$ and $y'' = y'''$.

- Assume $x \precsim y$ and $y \precsim x$ are witnessed by relations $R_1 \subseteq X \times X$ and $R_2 \subseteq X \times X$, respectively. We define $R := R_1 \cap R_2^r$, and show that R is a bisimulation witnessing $x \simeq y$. Clearly xR_1y and xR_2^ry, i.e. xRy. Thus let $x'Ry'$ for arbitrary $x', y' \in X$, then we find:
  - If $\delta(x')(\alpha) = 0$, then $\delta(y')(\alpha) = 0$. Indeed, if $\delta(y')(\alpha) = 1$ or $\delta(y')(\alpha) \notin 2$, we could falsely deduce $\delta(x')(\alpha) = 1$ or $\delta(x')(\alpha) \notin 2$, as $y'R_2x'$, and R_2 is a simulation.
  - If $\delta(x')(\alpha) = 1$, then $\delta(y')(\alpha) = 1$, since $x'R_1y'$, and R_1 is a simulation.
  - If $\delta(x')(\alpha) = (p, x'')$, then (i) $\delta(y')(\alpha) = (p, y'')$ with $x''R_1y''$, since $x'R_1y'$, and R_1 is a simulation; and (ii) $y''R_2x''$, since $y'R_2x'$ implies $\delta(x')(\alpha) = (p, x''') = (p, x'')$ for $y''R_2x'''$. Thus we find by definition of R that $x''Ry''$.

□

Lemma 3.3.4.3. *If $x \precsim y$ then $[\![x]\!] \subseteq [\![y]\!]$.*

Proof. The proof is similar to the one of its bisimilar counterpart [142, Lemma 5.2]. We prove $w \in [\![x]\!]$ implies $w \in [\![y]\!]$ for all $w \in \mathsf{GS}$ by induction on the length of w.

- For the induction base, let $w = \alpha$, then:

$$\begin{aligned} \alpha \in [\![x]\!] &\Leftrightarrow \delta(x)(\alpha) = 1 && \text{(Definition of } [\![-]\!]) \\ &\Rightarrow \delta(y)(\alpha) = 1 && (x \precsim y) \\ &\Leftrightarrow \alpha \in [\![y]\!] && \text{(Definition of } [\![-]\!]) \end{aligned}$$

- For the induction step, let $w = \alpha p v$, then we derive:

$$\begin{aligned} \alpha p v \in [\![x]\!] &\Leftrightarrow \delta(x)(\alpha) = (p, x'),\ v \in [\![x']\!] && \text{(Definition of } [\![-]\!]) \\ &\Rightarrow \delta(y)(\alpha) = (p, y'),\ v \in [\![y']\!] && (x \precsim y,\ \text{IH}) \\ &\Leftrightarrow \alpha p v \in [\![y]\!] && \text{(Definition of } [\![-]\!]) \end{aligned}$$

□

Lemma 3.3.4.4. *Let $L_1, L_2 \in \mathscr{L}$, then $L_1 \subseteq L_2$ iff $L_1 \precsim L_2$ in $(\mathscr{L}, \delta^{\mathscr{L}})$.*

Proof. Lemma 3.3.4.3 shows that $L_1 \precsim L_2$ implies $L_1 = [\![L_1]\!] \subseteq [\![L_2]\!] = L_2$.

Conversely, we show that $\subseteq$ is a simulation. Assume $L_1 \subseteq L_2$, then we compute:

$$\begin{aligned}
\delta^{\mathscr{L}}(L_1)(\alpha) = 1 &\Leftrightarrow \alpha \in L_1 && \text{(Definition of } \delta^{\mathscr{L}}\text{)} \\
&\Rightarrow \alpha \in L_2 && (L_1 \subseteq L_2) \\
&\Leftrightarrow \delta^{\mathscr{L}}(L_2)(\alpha) = 1 && \text{(Definition of } \delta^{\mathscr{L}}\text{)}
\end{aligned}$$

Moreover, we find:

$$\begin{aligned}
\delta^{\mathscr{L}}(L_1)(\alpha) = (p, L^1) &\Leftrightarrow \emptyset \neq L^1 = (\alpha p)^{-1} L_1 && \text{(Definition of } \delta^{\mathscr{L}}\text{)} \\
&\Rightarrow \emptyset \neq L^1 = (\alpha p)^{-1} L_1 \subseteq (\alpha p)^{-1} L_2 = L^2 && (L_1 \subseteq L_2) \\
&\Leftrightarrow \delta^{\mathscr{L}}(L_2)(\alpha) = (p, L^2),\ L^1 \subseteq L^2 && \text{(Definition of } \delta^{\mathscr{L}}\text{)}
\end{aligned}$$

□

Corollary 3.3.4.5. *Let $\mathscr{X}$ be a normal G-coalgebra, then $x \precsim y$ iff $[\![x]\!] \subseteq [\![y]\!]$.*

Proof. The proof is similar to the one of its bisimilar counterpart [142, Corollary 5.9].

From Lemma 3.3.4.3 it follows that $x \precsim y$ implies $[\![x]\!] \subseteq [\![y]\!]$.

Conversely, assume $[\![x]\!] \subseteq [\![y]\!]$. We define a relation $R := \{(s,t) \in X \times X \mid [\![s]\!] \subseteq [\![t]\!]\}$. In order to show $x \precsim y$ it is sufficient to prove that R is a simulation. Since $\mathscr{X}$ is normal, $[\![-]\!]$ is a G-coalgebra homomorphism.

- Suppose sRt and $\delta(s)(\alpha) = 1$. As $[\![-]\!]$ is a G-coalgebra homomorphism it follows $\delta^{\mathscr{L}}([\![s]\!])(\alpha) = 1$. Since $[\![s]\!] \subseteq [\![t]\!]$ implies $[\![s]\!] \precsim [\![t]\!]$ by Lemma 3.3.4.4, we thus can deduce $\delta^{\mathscr{L}}([\![t]\!])(\alpha) = 1$. Since $[\![-]\!]$ is a G-coalgebra homomorphism, we can conclude $\delta(t)(\alpha) = 1$.

- Suppose sRt and $\delta(s)(\alpha) = (p, s')$. Since $[\![-]\!]$ is a G-coalgebra homomorphism it follows $\delta^{\mathscr{L}}([\![s]\!])(\alpha) = (p, [\![s']\!])$. Since $[\![s]\!] \subseteq [\![t]\!]$ implies $[\![s]\!] \precsim [\![t]\!]$ by Lemma 3.3.4.4, we deduce $\delta^{\mathscr{L}}([\![t]\!])(\alpha) = (p, L)$ for some $L \in \mathscr{L}$ with $[\![s']\!] \precsim L$ in $\mathscr{L}$. Since $[\![-]\!]$ is a G-coalgebra homomorphism, it follows $L = [\![t']\!]$ with $\delta(t)(\alpha) = (p, t')$. Thus we have $[\![s']\!] \precsim [\![t']\!]$, or equivalently $[\![s']\!] \subseteq [\![t']\!]$ by Lemma 3.3.4.4. The latter implies $s'Rt'$ by definition of R. Thus we find $\delta(t)(\alpha) = (p, t')$ and $s'Rt'$. □

3.4 The Minimal Representation $m(\mathscr{X})$

The automaton $\mathscr{X}_e$ assigned to an expression e by the Thompson construction is not always the most efficient representation of the language $[\![e]\!]$. For instance, as seen in Example 3.3.3.1, the Thompson-automaton $\mathscr{X}_{p^{(b)}q}$ in Figure 3.4 contains redundant structure, since its states x and y exhibit the same behaviour. In this section, we show that any G-automaton $\mathscr{X}$ admits an equivalent *minimal* representation, $m(\mathscr{X})$.

3.4.1 Reachability

We begin by formally defining what it means for a state of a G-automaton to be reachable, and show that restricting an automaton to its reachable states leaves important properties invariant.

Definition 3.4.1.1. Let (X, δ) be a G-coalgebra. We write $\to \subseteq X \times \mathsf{GS}^- \times X$ for the smallest relation satisfying:

$$\frac{}{x \xrightarrow{\varepsilon} x} \qquad \frac{\delta(x)(\alpha) = (p, y)}{x \xrightarrow{\alpha p} y} \qquad \frac{x \xrightarrow{\alpha_1 p_1 \ldots \alpha_{n-1} p_{n-1}} y \quad , \quad y \xrightarrow{\alpha_n p_n} z}{x \xrightarrow{\alpha_1 p_1 \ldots \alpha_n p_n} z}. \tag{3.3}$$

The states *reachable* from $x \in X$ are $r(x) := \{y \in X \mid \exists w \in \mathsf{GS}^- : x \xrightarrow{w} y\}$, and their *witnesses* are $R(x) := \{w \in \mathsf{GS}^- \mid \exists x_w \in X : x \xrightarrow{w} x_w\}$.

The following result shows that a state reached by a word is uniquely defined.

Lemma 3.4.1.2. *If $x \xrightarrow{w} x^1_w$ and $x \xrightarrow{w} x^2_w$, then $x^1_w = x^2_w$.*

Proof. We show the statement by induction on the length of $w \in \mathsf{GS}^-$:

- The induction base $w = \varepsilon$ follows from the base case of (3.3): $x \xrightarrow{\varepsilon} x^i_w$ iff $x^i_w = x$ for $i = 1, 2$.
- For the induction step let $w = v\alpha p$ for some $v \in \mathsf{GS}^-$. By (3.3) there exist $x^1_v, x^2_v \in X$ such that $x \xrightarrow{v} x^1_v \xrightarrow{\alpha p} x^1_w$ and $x \xrightarrow{v} x^2_v \xrightarrow{\alpha p} x^2_w$. From the induction hypothesis it follows $x^1_v = x^2_v$. Thus, by (3.3), $(p, x^1_w) = \delta(x^1_v)(\alpha) = \delta(x^2_v)(\alpha) = (p, x^2_w)$, which yields $x^1_w = x^2_w$. □

Given a G-coalgebra (X, δ), we call a subset $A \subseteq X$ *δ-invariant*, if $y \in A$ and $\delta(y)(\alpha) = (p, z)$, then $z \in A$. In such a situation, we write $\mathscr{X}^A = (A, \delta^A)$ for the well-defined restriction of $\mathscr{X} = (X, \delta)$ to A.

We denote the sub-automaton one obtains by restricting a G-automaton $\mathscr{X} = (X, \delta, x)$ to the δ-invariant set of states reachable from the initial state $x \in X$ by $r(\mathscr{X}) := \mathscr{X}^{r(x)}$, and call an automaton *reachable*, if $\mathscr{X} = r(\mathscr{X})$. Following [65, Def. 15], we call a normal, reachable, and observable automaton *minimal*.

The set $R(x)$ of words witnessing the reachability of states in $\mathscr{X} = (X, \delta, x)$ can be equipped with a G-automaton structure $R(\mathscr{X}) := (R(x), \partial, \varepsilon)$, where $\partial(w)(\alpha) = (p, w\alpha p)$, if $\delta(x_w)(\alpha) = (p, x_{w\alpha p})$ for some $x_{w\alpha p} \in X$, and $\partial(w)(\alpha) = \delta(x_w)(\alpha)$ otherwise. The automaton $r(\mathscr{X})$ can then be recovered as the image of the automata homomorphism $f : R(\mathscr{X}) \to \mathscr{X}$ defined by $f(w) = x_w$. In other words, there exists an epi-mono factorisation $R(\mathscr{X}) \twoheadrightarrow r(\mathscr{X}) \hookrightarrow \mathscr{X}$.

We conclude with a list of important properties preserved by restricting an automaton to its reachable states. Among those properties are *well-nestedness* [142] and *satisfing the nesting coequation* [135]. We refer the reader to Section 4.7 for a high-level comparison between the two notions.

The definition of well-nestedness requires the following construction.

Definition 3.4.1.3 ([142])**.** Let $\mathscr{X} = (X, \delta)$ be a G-coalgebra. The *uniform continuation* of $A \subseteq X$ by $h \in G(X)$ is the G-coalgebra $\mathscr{X}[A, h] = (X, \delta[A, h])$, where:

$$\delta[A, h](x)(\alpha) = \begin{cases} h(\alpha) & \text{if } x \in A, \delta(x)(\alpha) = 1 \\ \delta(x)(\alpha) & \text{else} \end{cases}.$$

Further let $\mathscr{X} + \mathscr{Y} := (X + Y, \delta^{\mathscr{X}} + \delta^{\mathscr{Y}})$ be the disjoint union of automata, where $\delta^{\mathscr{X}} + \delta^{\mathscr{Y}}(x)(\alpha) = \delta^{\mathscr{X}}(x)(\alpha)$, if $x \in X$, and $\delta^{\mathscr{Y}}(x)(\alpha)$ otherwise.

Definition 3.4.1.4 ([142])**.** The class of *well-nested* G-coalgebras is defined as follows:

- If $\mathscr{X} = (X, \delta)$ has no transitions, i.e. if $\delta : X \to 2^{\mathsf{At}}$, then $\mathscr{X}$ is well-nested.
- If $\mathscr{X}$ and $\mathscr{Y}$ are well-nested and $h \in G(X + Y)$, then $(\mathscr{X} + \mathscr{Y})[X, h]$ is well-nested.

While the following results are stated in terms of arbitrary δ-invariant subsets, we are particularly interested in $r(x)$, the subset of states reachable by an initial state x.

Lemma 3.4.1.5. *The restriction of a well-nested G-coalgebra to a δ-invariant subset is well-nested.*

Proof. We show the statement by induction on the well-nested structure of $\mathscr{X}$. As before, we write $\mathscr{X}^A = (A, \delta^A)$ for the well-defined restriction of $\mathscr{X} = (X, \delta)$ to a δ-invariant subset $A \subseteq X$.

- For the induction base assume that $\mathscr{X} = (X, \delta)$ satisfies $\delta : X \to 2^{\mathsf{At}}$, and $A \subseteq X$ is a δ-invariant set. Then clearly the restriction is of type $\delta^A : A \to 2^{\mathsf{At}}$, i.e. $\mathscr{X}^A = (A, \delta^A)$ is well-nested.

- For the induction step let $\mathscr{Y} = (Y, \delta^{\mathscr{Y}})$ and $\mathscr{Z} = (Z, \delta^{\mathscr{Z}})$ be well-nested G-coalgebras, $h \in G(Y + Z)$, and $\mathscr{X} = (Y + Z, (\delta^{\mathscr{Y}} + \delta^{\mathscr{Z}})[Y, h])$. Moreover let $A \subseteq Y + Z$ be a $(\delta^{\mathscr{Y}} + \delta^{\mathscr{Z}})[Y, h]$-invariant set. We would like to show that $\mathscr{X}^A$ is well-nested. The induction hypothesis reads:

  - for all $\delta^{\mathscr{Y}}$-invariant sets $B \subseteq Y$, the subcoalgebra $\mathscr{Y}^B = (B, (\delta^{\mathscr{Y}})^B)$ is well-nested;
  - for all $\delta^{\mathscr{Z}}$-invariant sets $C \subseteq Z$, the subcoalgebra $\mathscr{Z}^C = (C, (\delta^{\mathscr{Z}})^C)$ is well-nested.

  We begin by showing that $A \cap Y \subseteq Y$ and $A \cap Z \subseteq Z$ are $\delta^{\mathscr{Y}}$- and $\delta^{\mathscr{Z}}$-invariant sets, respectively. Let $\delta^{\mathscr{Y}}(x)(\alpha) = (p, y)$ for $x \in A \cap Y$ and $y \in Y$. Then by definition

  $$(\delta^{\mathscr{Y}} + \delta^{\mathscr{Z}})[Y, h](x)(\alpha) = \delta^{\mathscr{Y}}(x)(\alpha) = (p, y),$$

  which by $(\delta^{\mathscr{Y}} + \delta^{\mathscr{Z}})[Y, h]$-invariance of A implies $y \in A$. It hence follows $y \in A \cap Y$. Analogously one dedues that $A \cap Z$ is $\delta^{\mathscr{Z}}$-invariant. Thus $\mathscr{Y}^{A \cap Y} = (A \cap Y, (\delta^{\mathscr{Y}})^{A \cap Y})$ and $\mathscr{Z}^{A \cap Z} = (A \cap Z, (\delta^{\mathscr{Z}})^{A \cap Z})$ are well-defined, and moreover, by the induction hypothesis they are well-nested.

  We observe the equality $A = A \cap Y + A \cap Z$, which follows from $A \subseteq Y + Z$,

and define $\overline{h} \in G(A \cap Y + A \cap Z) = G(A)$ by:

$$\overline{h}(\alpha) = \begin{cases} 1 & \text{if } h(\alpha) = 1 \\ (p, x) & \text{if } h(\alpha) = (p, x),\ x \in A \\ 0 & \text{else} \end{cases} \tag{3.4}$$

It follows $\mathscr{X}^A = (A \cap Y + A \cap Z, ((\delta^{\mathscr{Y}})^{A \cap Y} + (\delta^{\mathscr{Z}})^{A \cap Z})[A \cap Y, \overline{h}])$, since for any $x \in A$ it holds:

$$\begin{aligned}
&((\delta^{\mathscr{Y}} + \delta^{\mathscr{Z}})[Y, h])^A(x)(\alpha) \\
&= \begin{cases} \delta^{\mathscr{Y}}(x)(\alpha) & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) \neq 1 \\ h(\alpha) & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) = 1 \\ \delta^{\mathscr{Z}}(x)(\alpha) & x \in A \cap Z \end{cases} \\
&= \begin{cases} \delta^{\mathscr{Y}}(x)(\alpha) & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) \neq 1 \\ 1 & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) = 1,\ h(\alpha) = 1 \\ 0 & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) = 1,\ h(\alpha) = 0 \\ (p, x') & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) = 1,\ h(\alpha) = (p, x') \\ \delta^{\mathscr{Z}}(x)(\alpha) & x \in A \cap Z \end{cases} \\
&\overset{(\star)}{=} \begin{cases} \delta^{\mathscr{Y}}(x)(\alpha) & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) \neq 1 \\ 1 & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) = 1, \overline{h}(\alpha) = 1 \\ 0 & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) = 1, \overline{h}(\alpha) = 0 \\ (p, x') & x \in A \cap Y,\ \delta^{\mathscr{Y}}(x)(\alpha) = 1, \overline{h}(\alpha) = (p, x') \\ \delta^{\mathscr{Z}}(x)(\alpha) & x \in A \cap Z \end{cases} \\
&= \begin{cases} (\delta^{\mathscr{Y}})^{A \cap Y}(x)(\alpha) & x \in A \cap Y,\ (\delta^{\mathscr{Y}})^{A \cap Y}(x)(\alpha) \neq 1 \\ \overline{h}(\alpha) & x \in A \cap Y,\ (\delta^{\mathscr{Y}})^{A \cap Y}(x)(\alpha) = 1 \\ (\delta^{\mathscr{Z}})^{A \cap Z}(x)(\alpha) & x \in A \cap Z \end{cases} \\
&= ((\delta^{\mathscr{Y}})^{A \cap Y} + (\delta^{\mathscr{Z}})^{A \cap Z})[A \cap Y, \overline{h}](x)(\alpha),
\end{aligned}$$

where we use for $(\star)$ that $((\delta^{\mathscr{Y}} + \delta^{\mathscr{Z}})[Y,h])^A(x)(\alpha) = (p,x')$ for $x \in A$, implies $x' \in A$, as A is $(\delta^{\mathscr{Y}} + \delta^{\mathscr{Z}})[Y,h]$-invariant. □

Restricting to a δ-invariant subset not only preserves well-nestedness, but also language semantics.

Lemma 3.4.1.6. *Let $\mathscr{X}^A = (A, \delta^A)$ be the restriction of a G-coalgebra $\mathscr{X} = (X, \delta)$ to a δ-invariant subset $A \subseteq X$. Then $[\![a]\!]_{\mathscr{X}} = [\![a]\!]_{\mathscr{X}^A}$ for all $a \in A$.*

Proof. We show $w \in [\![a]\!]_{\mathscr{X}}$ iff $w \in [\![a]\!]_{\mathscr{X}^A}$ for all $a \in A$ and $w \in \mathsf{GS}$ by induction on the length of w.

- For the induction base assume $w = \alpha$, then we deduce:

$$\begin{aligned} \alpha \in [\![a]\!]_{\mathscr{X}} &\Leftrightarrow \delta(a)(\alpha) = 1 && \text{(Definition of } [\![-]\!]) \\ &\Leftrightarrow \delta^A(a)(\alpha) = 1 && (a \in A) \\ &\Leftrightarrow \alpha \in [\![a]\!]_{\mathscr{X}^A} && \text{(Definition of } [\![-]\!]). \end{aligned}$$

- For the induction step let $w = \alpha p v$, then we find:

$$\begin{aligned} \alpha p v \in [\![a]\!]_{\mathscr{X}} &\Leftrightarrow \exists x \in X : \delta(a)(\alpha) = (p,x),\ v \in [\![x]\!]_{\mathscr{X}} && \text{(Definition of } [\![-]\!]) \\ &\Leftrightarrow \exists b \in A : \delta(a)(\alpha) = (p,b),\ v \in [\![b]\!]_{\mathscr{X}} && (a \in A,\ \delta\text{-inv}) \\ &\Leftrightarrow \exists b \in A : \delta^A(a)(\alpha) = (p,b),\ v \in [\![b]\!]_{\mathscr{X}^A} && (a,b \in A,\ \text{IH}) \\ &\Leftrightarrow \alpha p v \in [\![a]\!]_{\mathscr{X}^A} && \text{(Definition of } [\![-]\!]). \end{aligned}$$

□

In consequence, we immediately obtain that restricting to a δ-invariant subset preserves normality.

Lemma 3.4.1.7. *The restriction of a normal G-coalgebra to a δ-invariant subset is normal.*

Proof. Let $\mathscr{X} = (X, \delta)$ be a normal G-coalgebra and $A \subseteq X$ a δ-invariant subset. We write $\mathscr{X}^A = (A, \delta^A)$ for the restriction of $\mathscr{X}$ to A. Assume for $a, b \in A$ we have

$\delta^A(a)(\alpha) = (p, b)$. Since $a \in A$, we have $\delta(a)(\alpha) = (p, b)$, which by normality of $\mathscr{X}$ implies $\emptyset \neq [\![b]\!]_{\mathscr{X}}$. From $b \in A$ and Lemma 3.4.1.6 we thus can deduce $\emptyset \neq [\![b]\!]_{\mathscr{X}A}$. □

We will conclude this section with a summarising result. We say that a G-automaton $\mathscr{X}$ *satisfies the nesting coequation*, if the final coalgebra homomorphism $\mathsf{obs}_{\mathscr{X}} = [\![\cdot]\!] : \mathscr{X} \to \mathcal{L}$ factors through the *coequation* $\{[\![e]\!] \mid e \in \mathsf{Expr}\}$ – that is, for any $x \in X$ there exists an expression $e_x \in \mathsf{Expr}$ such that $[\![e_x]\!] = [\![x]\!]$. The interested reader will find more details in [46, 135]. For our purposes it is sufficient to know that the class of all G-automata satisfying the nesting coequation forms a *covariety* [135]. Covarieties are a categorical dualization of *varieties*, which are well-known from universal algebra (cf. Section 5.3.6). Birkhoff's famous HSP theorem states that varieties are closed under homomorphic images (H), subalgebras (S), and products (P) [29]. Covarieties enjoy similarly desirable properties: they are closed under homomorphic images, subcoalgebras, and coproducts [46].

Proposition 3.4.1.8. *Let $\mathscr{X}$ be a G-automaton, then $r(\mathscr{X})$ is well-nested, normal, or satisfies the nesting coequation, whenever $\mathscr{X}$ does. Moreover, $r(\mathscr{X})$ accepts the same language as $\mathscr{X}$.*

Proof. Let us write $\mathscr{X} = (X, \delta, x)$. From (3.3) it is immediate that $r(x) \subseteq X$ is δ-invariant. Thus, Lemma 3.4.1.5 and Lemma 3.4.1.7, respectively, imply that $r(\mathscr{X}) = (r(x), \delta^{r(x)}, x)$ is well-nested, or normal, whenever $\mathscr{X}$ is. From Lemma 3.4.1.6 it further follows

$$[\![r(\mathscr{X})]\!] = [\![x]\!]_{r(\mathscr{X})} = [\![x]\!]_{\mathscr{X}} = [\![\mathscr{X}]\!].$$

Since the class of all G-automata satisfying the nesting coequation forms a covariety, it is closed under subautomata. As there exists an epi-mono factorisation

$$R(\mathscr{X}) \twoheadrightarrow r(\mathscr{X}) \hookrightarrow \mathscr{X}$$

the automaton $r(\mathscr{X})$ thus satisfies the nesting coequation, whenever $\mathscr{X}$ does. □

3.4.2 Minimality

Recall that the state-space of the minimal DFA for a regular language consists of the equivalence classes of the Myhill-Nerode equivalence relation [120].

Similarly, we define the state-space of the minimisation of a GKAT automaton $\mathscr{X}$ as the equivalence classes of the equivalence relation $\equiv_{[\![\mathscr{X}]\!]}$ on GS^-, defined for any guarded string language $L \subseteq \mathsf{GS}$ by:

$$v \equiv_L w :\Leftrightarrow \forall u \in \mathsf{GS} : vu \in L \text{ iff } wu \in L. \tag{3.5}$$

Let $v^{-1}L = \{u \in \mathsf{GS} \mid vu \in L\}$ be the derivative of L with respect to v. Then two words v, w are equivalent with respect to $\equiv_L$ iff the derivatives $v^{-1}L$ and $w^{-1}L$ coincide.

Definition 3.4.2.1. The *minimisation* of a G-automaton $\mathscr{X} = (X, \delta, x)$ is $m(\mathscr{X}) := (\{w^{-1}[\![\mathscr{X}]\!] \mid w \in R(x)\}, \partial, [\![\mathscr{X}]\!])$ with:

$$\partial(L)(\alpha) := \begin{cases} (p, (\alpha p)^{-1}L) & \text{if } (\alpha p)^{-1}L \neq \emptyset \\ 1 & \text{if } \alpha \in L \\ 0 & \text{otherwise} \end{cases}, \tag{3.6}$$

for $L \in \{w^{-1}[\![\mathscr{X}]\!] \mid w \in R(x)\}$.

A few remarks on the well-definedness of above definition are in order. The language accepted by a G-automaton is deterministic, and taking the derivative of a language preserves its deterministic nature. Thus only one of the three cases in (3.6) occurs. Since $\varepsilon \in R(x)$ and $\varepsilon^{-1}L = L$, the initial state of the minimisation is well-defined. Transitioning to a new state is well-defined since $v^{-1}(w^{-1}L) = (wv)^{-1}L$.

It is not hard to see that on a high-level the minimisation can be recovered as the image of the final automata homomorphism $[\![-]\!] : R(\mathscr{X}) \to \mathcal{L}$, which, as the result below shows, satisfies $[\![w]\!]_{R(\mathscr{X})} = w^{-1}[\![\mathscr{X}]\!]$.

Lemma 3.4.2.2. *Let $\mathscr{X}$ be a G-automaton with initial state $x \in X$. Then $[\![w]\!]_{R(\mathscr{X})} = w^{-1}[\![\mathscr{X}]\!]$ for all $w \in R(x)$.*

Proof. We prove $u \in [\![w]\!]_{R(\mathscr{X})}$ iff $u \in w^{-1}[\![\mathscr{X}]\!]$ for all $u \in \mathsf{GS}$ and $w \in R(x)$ by induction on the length of u.

- For the induction base assume $u = \alpha$, then we find:

$$\begin{aligned}
\alpha \in [\![w]\!]_{R(\mathscr{X})} &\Leftrightarrow \partial(w)(\alpha) = 1 && \text{(Definition of } [\![-]\!]) \\
&\Leftrightarrow \delta(x_w)(\alpha) = 1 && \text{(Definition of } \partial) \\
&\Leftrightarrow \alpha \in [\![x_w]\!]_{\mathscr{X}} && \text{(Definition of } [\![-]\!]) \\
&\Leftrightarrow w\alpha \in [\![\mathscr{X}]\!] && \text{(Definition of } [\![-]\!]) \\
&\Leftrightarrow \alpha \in w^{-1}[\![\mathscr{X}]\!] && \text{(Definition of } w^{-1}[\![\mathscr{X}]\!])
\end{aligned}$$

- For the induction step let $u = \alpha p v$, then it follows:

$$\begin{aligned}
& \alpha p v \in [\![w]\!]_{R(\mathscr{X})} \\
\Leftrightarrow\ & \partial(w)(\alpha) = (p, w\alpha p),\ v \in [\![w\alpha p]\!]_{R(\mathscr{X})} && \text{(Definition of } [\![-]\!]) \\
\Leftrightarrow\ & \delta(x_w)(\alpha) = (p, x_{w\alpha p}),\ v \in (w\alpha p)^{-1}[\![\mathscr{X}]\!] && \text{(Definition of } \partial,\ \text{IH}) \\
\Leftrightarrow\ & \delta(x_w)(\alpha) = (p, x_{w\alpha p}),\ v \in [\![x_{w\alpha p}]\!]_{\mathscr{X}} && \text{(Definition of } (w\alpha p)^{-1}[\![\mathscr{X}]\!]) \\
\Leftrightarrow\ & \alpha p v \in [\![x_w]\!]_{\mathscr{X}} && \text{(Definition of } [\![-]\!]) \\
\Leftrightarrow\ & \alpha p v \in w^{-1}[\![\mathscr{X}]\!] && \text{(Definition of } w^{-1}[\![\mathscr{X}]\!])
\end{aligned}$$

□

In other words, there exists an epi-mono factorisation $R(\mathscr{X}) \twoheadrightarrow m(\mathscr{X}) \hookrightarrow \mathcal{L}$.

Properties of $m(\mathscr{X})$

In this section we prove properties of $m(\mathscr{X})$, which one would expect to hold by a minimisation construction. We begin by showing that minimising a normal automaton results in a reachable acceptor.

Lemma 3.4.2.3. *Let $\mathscr{X}$ be a normal G-automaton with initial state $x \in X$. Then $[\![\mathscr{X}]\!] \xrightarrow{w} w^{-1}[\![\mathscr{X}]\!]$ in $m(\mathscr{X})$ for all $w \in R(x)$. In particular, $m(\mathscr{X})$ is reachable.*

Proof. We prove the statement by induction on the length of $w \in R(x)$:

- For the induction base, let $w = \varepsilon$, then $[\![\mathscr{X}]\!] \xrightarrow{\varepsilon} [\![\mathscr{X}]\!] = \varepsilon^{-1}[\![\mathscr{X}]\!]$ by the base case of (3.3).

- In the induction step, let $w = v\alpha p$ with $v \in \mathsf{GS}^-$. By the definition of reachability, $v \in R(x)$. From the induction hypothesis we deduce $[\![\mathscr{X}]\!] \xrightarrow{v} v^{-1}[\![\mathscr{X}]\!]$ in $m(\mathscr{X})$. The normality of $\mathscr{X}$ implies the inequality $(\alpha p)^{-1}(v^{-1}[\![\mathscr{X}]\!]) \neq \emptyset$. From (3.6) it thus follows $v^{-1}[\![\mathscr{X}]\!] \xrightarrow{\alpha p} (\alpha p)^{-1}(v^{-1}[\![\mathscr{X}]\!]) = w^{-1}[\![\mathscr{X}]\!]$. We conclude $[\![\mathscr{X}]\!] \xrightarrow{w=v\alpha p} w^{-1}[\![\mathscr{X}]\!]$ by (3.3). □

The next result proves that minimisation preserves language semantics.

Lemma 3.4.2.4. *Let $\mathscr{X}$ be a G-automaton, then $[\![L]\!] = L$ for all L in $m(\mathscr{X})$. In particular, $[\![m(\mathscr{X})]\!] = [\![\mathscr{X}]\!]$.*

Proof. We show $v \in [\![w^{-1}[\![\mathscr{X}]\!]]\!]$ iff $v \in w^{-1}[\![\mathscr{X}]\!]$ for all $v \in \mathsf{GS}$, $w \in R(x)$, by induction on the length of v:

- For the induction base, let $v = \varepsilon$. Then we can compute:

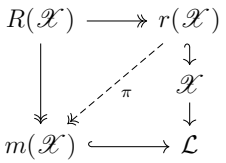
$$\begin{aligned} \alpha \in [\![w^{-1}[\![\mathscr{X}]\!]]\!] &\Leftrightarrow \partial(w^{-1}[\![\mathscr{X}]\!])(\alpha) = 1 && \text{(Definition of } [\![-]\!]) \\ &\Leftrightarrow \alpha \in w^{-1}[\![\mathscr{X}]\!] && \text{(Definition of } \partial) \end{aligned}$$

- In the induction step, let $v = \alpha p u$. Then we have the following equivalences:

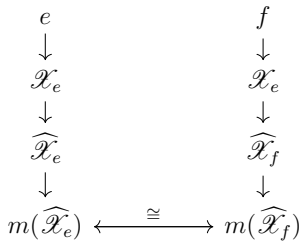
$$\begin{aligned} \alpha p u \in [\![w^{-1}[\![\mathscr{X}]\!]]\!] &\Leftrightarrow u \in [\![(w\alpha p)^{-1}[\![\mathscr{X}]\!]]\!] && \text{(Definition of } [\![-]\!], (3.6)) \\ &\Leftrightarrow u \in (w\alpha p)^{-1}[\![\mathscr{X}]\!] && \text{(IH)} \\ &\Leftrightarrow w\alpha p u \in [\![\mathscr{X}]\!] && \text{(Definition of } (-)^{-1}[\![\mathscr{X}]\!]) \\ &\Leftrightarrow \alpha p u \in w^{-1}[\![\mathscr{X}]\!] && \text{(Definition of } (-)^{-1}[\![\mathscr{X}]\!]) \end{aligned}$$

In particular, $[\![m(\mathscr{X})]\!] = [\![[\![\mathscr{X}]\!]]\!] = [\![\varepsilon^{-1}[\![\mathscr{X}]\!]]\!] = \varepsilon^{-1}[\![\mathscr{X}]\!] = [\![\mathscr{X}]\!]$. □

An immediate consequence of above statement is that the states of the minimisation can be distinguished by their observable behaviour, that is, different states accept different languages. Another implication of Lemma 3.4.2.4 is the normality of the minimisation: all states are *live*.



(a) The morphism π as unique diagonal



(b) $[\![e]\!] = [\![f]\!]$ iff $m(\widehat{\mathscr{X}_e})$ and $m(\widehat{\mathscr{X}_f})$ are isomorphic

Figure 3.5: A high-level view of the notions introduced in Section 3.4.2

Corollary 3.4.2.5. *Let $\mathscr{X}$ be a G-automaton, then $m(\mathscr{X})$ is normal and observable.*

Proof.

- By Lemma 3.4.2.4, $[\![L_1]\!] = [\![L_2]\!]$ implies $L_1 = L_2$, which shows that $[\![-]\!]$ is injective. By definition, this proves that $m(\mathscr{X})$ is observable.
- Assume $\partial(L_1)(\alpha) = (p, L_2)$, then $L_2 = (\alpha p)^{-1} L_1 \neq \emptyset$ by (3.6). It thus follows from Lemma 3.4.2.4 that $[\![L_2]\!] = L_2 \neq \emptyset$, which shows that $m(\mathscr{X})$ is normal. □

Since $m(\mathscr{X})$ is normal, reachable, and observable, if $\mathscr{X}$ is normal, it is, by our definition, *minimal* (cf. [65, Def. 15]). Its size-minimality among normal automata language equivalent to $\mathscr{X}$ follows from the abstract definition, cf. Corollary 3.4.2.10.

Identifying $m(\mathscr{X})$

In this section, we identify the minimisation of a normal G-automaton with an alternative, but equivalent, construction. In consequence, we are able to derive that the minimisation of a normal automaton is size-minimal among language equivalent normal automata and preserves the nesting coequation. We begin by observing its universality in the following sense.

Proposition 3.4.2.6. *Let $\mathscr{X}$ and $\mathscr{Y}$ be normal G-automata with $[\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$, and $y \in Y$ the initial state of $\mathscr{Y}$. Then $\pi : r(\mathscr{Y}) \to m(\mathscr{X})$ with $\pi(z) = w_z^{-1}[\![\mathscr{X}]\!]$, for $y \xrightarrow{w_z} z$ in $\mathscr{Y}$, is a (surjective) G-automata homomorphism, uniquely defined.*

Proof. We have to show that π is well-defined, surjective, preserves initial states, is a G-coalgebra homomorphism, and is unique. In this order:

- Let $z \in r(y)$, then by definition there exists at least one $w_1 \in R(y)$ such that $y \xrightarrow{w_1} z$ in $\mathscr{Y}$. Since $\mathscr{Y}$ is normal, we have $w_1^{-1}[\![\mathscr{X}]\!] = w_1^{-1}[\![\mathscr{Y}]\!] \neq \emptyset$. Hence there exists some $z' \in X$, such that $x \xrightarrow{w_1} z'$ in $\mathscr{X}$, that is, $w_1 \in R(x)$, where x is the initial state of $\mathscr{X}$. Assume there exists a second $w_2 \in R(y)$, such that $y \xrightarrow{w_2} z$ in $\mathscr{Y}$. Then we have:

$$\begin{aligned} w_1 u \in [\![\mathscr{X}]\!] &\Leftrightarrow w_1 u \in [\![\mathscr{Y}]\!] && ([\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]) \\ &\Leftrightarrow u \in [\![z]\!] && (\text{Definition of } [\![-]\!]) \\ &\Leftrightarrow w_2 u \in [\![\mathscr{Y}]\!] && (\text{Definition of } [\![-]\!]) \\ &\Leftrightarrow w_2 u \in [\![\mathscr{X}]\!] && ([\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]) \end{aligned}$$

  for all $u \in \mathsf{GS}$. In other words, $w_1 \equiv_{[\![\mathscr{X}]\!]} w_2$, or, equivalently, $w_1^{-1}[\![\mathscr{X}]\!] = w_2^{-1}[\![\mathscr{X}]\!]$. Thus π is a well-defined function.

- Let $w \in R(x)$, then by definition there exists $x_w \in X$ with $x \xrightarrow{w} x_w$ in $\mathscr{X}$. Since $\mathscr{X}$ is normal, $w^{-1}[\![\mathscr{Y}]\!] = w^{-1}[\![\mathscr{X}]\!] \neq \emptyset$, i.e. $y \xrightarrow{w} y_w$ in $\mathscr{Y}$, for some $y_w \in Y$. Thus, by construction, $\pi(y_w) = w^{-1}[\![\mathscr{X}]\!]$, which shows that π is surjective.

- Initial states are preserved since by (3.3) we have $y \xrightarrow{\varepsilon} y$, which by definition of π implies $\pi(y) = \varepsilon^{-1}[\![\mathscr{X}]\!] = [\![\mathscr{X}]\!]$.

- π is a G-coalgebra homomorphism:

  - Let $\delta^{\mathscr{Y}}(z)(\alpha) = 0$, then $\delta^{m(\mathscr{X})}(\pi(z))(\alpha) \neq 1$, since otherwise $w_z\alpha \in [\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$ by the definition of $\delta^{m(\mathscr{X})}$, which would imply the contradiction $1 = \delta^{\mathscr{Y}}(z)(\alpha) = 0$. Assume $\delta^{m(\mathscr{X})}(\pi(z))(\alpha) = (p, (w_z\alpha p)^{-1}[\![\mathscr{X}]\!])$. By definition of $\delta^{m(\mathscr{X})}$ there exists some $v \in \mathsf{GS}$, such that $w_z\alpha pv \in [\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$. Hence it follows $\delta^{\mathscr{Y}}(z)(\alpha) \neq 0$ by the definition of $[\![-]\!]$, which is a contradiction. We can thus conclude $\delta^{m(\mathscr{X})}(\pi(z))(\alpha) = 0$
  - Let $\delta^{\mathscr{Y}}(z)(\alpha) = 1$, then $w_z\alpha \in [\![\mathscr{Y}]\!] = [\![\mathscr{X}]\!]$ by the definition of $[\![-]\!]$. From the definition of $\delta^{m(\mathscr{X})}$, it follows $\delta^{m(\mathscr{X})}(\pi(z))(\alpha) = 1$.
  - Let $\delta^{\mathscr{Y}}(z)(\alpha) = (p, z')$, then, by normality of $\mathscr{Y}$, there exists some $v \in [\![z']\!] \neq \emptyset$. The latter implies $w_z\alpha pv \in [\![\mathscr{Y}]\!] = [\![\mathscr{X}]\!]$. By the definitions of $\delta^{m(\mathscr{X})}$ and $w_{z'}$, it follows $\delta^{m(\mathscr{X})}(\pi(z))(\alpha) = (p, (w_z\alpha p)^{-1}[\![\mathscr{X}]\!]) = (p, \pi(z'))$.

- Let $g : r(\mathscr{Y}) \to m(\mathscr{X})$ be any G-automata homomorphism. Let $z \in r(y)$, then by definition there exists $w_z \in R(y)$, such that $y \xrightarrow{w_z} z$ in $\mathscr{Y}$, and thus in $r(\mathscr{Y})$. Since g is a G-automata homomorphism, it follows $[\![\mathscr{X}]\!] = g(y) \xrightarrow{w_z} g(z)$ in $m(\mathscr{X})$. By Lemma 3.4.2.3, on the other hand, we have $[\![\mathscr{X}]\!] \xrightarrow{w_z} w_z^{-1}[\![\mathscr{X}]\!]$ in $m(\mathscr{X})$. From Lemma 3.4.1.2 it thus follows $g(z) = w_z^{-1}[\![\mathscr{X}]\!] = \pi(z)$. □

The next result shows that the minimisation of a normal G-automaton is isomorphic to the automaton that arises by identifying semantically equivalent pairs among reachable states.

Lemma 3.4.2.7. *Let $\mathscr{X}$ be a normal G-automaton with initial state $x \in X$ and $\pi : r(\mathscr{X}) \twoheadrightarrow m(\mathscr{X})$ as in Proposition 3.4.2.6, then $y \simeq z$ iff $\pi(y) = \pi(z)$ for all $y, z \in r(x)$. Consequently, $m(\mathscr{X})$ is isomorphic to $r(\mathscr{X})/\simeq$.*

Proof. The statement follows from the following chain of equivalences:

$$\begin{aligned} \pi(y) = \pi(z) &\Leftrightarrow (w_y)^{-1}[\![\mathscr{X}]\!] = (w_z)^{-1}[\![\mathscr{X}]\!] && \text{(Definition of } \pi) \\ &\Leftrightarrow [\![y]\!] = [\![z]\!] && \text{(Definition of } (-)^{-1}[\![\mathscr{X}]\!]) \\ &\Leftrightarrow y \simeq z && (\mathscr{X} \text{ is normal}) \end{aligned}$$

□

On a high level, the automata homomorphism π can be recovered as the unique (surjective) diagonal making the diagram in Figure 3.5a commute.

In Proposition 3.4.1.8 it was noted that the reachable subautomaton $r(\mathscr{X})$ satisfies the nesting coequation, whenever $\mathscr{X}$ does. By Proposition 3.4.2.6 there exists an epimorphism $\pi : r(\mathscr{X}) \twoheadrightarrow m(\mathscr{X})$, if $\mathscr{X}$ is normal. Since coalgebras satisfying a coequation form a covariety, which is closed under homomorphic images [46, 135], we thus can deduce the following result.

Corollary 3.4.2.8. *Let $\mathscr{X}$ be a normal G-automaton, then $m(\mathscr{X})$ satisfies the nesting coequation, whenever $\mathscr{X}$ does.*

Proof. In Proposition 3.4.1.8 it was noted that the reachable subcoalgebra $r(\mathscr{X})$ satisfies the nesting coequation, whenever $\mathscr{X}$ does. By Proposition 3.4.2.6 there

exists an epimorphism $\pi : r(\mathscr{X}) \twoheadrightarrow m(\mathscr{X})$ for any normal automaton $\mathscr{X}$. The claim follows since coalgebras satisfying a coequation form a covariety, which is in particular closed under homomorphic images [46, 135]. □

We continue with the observation that two normal G-automata are language equivalent if and only if their minimisations are isomorphic. As depicted in Figure 3.5b, this implies that two expressions e and f are language equivalent if and only if the minimisations of their normalised Thompson automata are isomorphic. A similar idea occurs in Kozen's completeness proof for Kleene Algebra [88, Theorem 19].

Corollary 3.4.2.9. *Let $\mathscr{X}$ and $\mathscr{Y}$ be normal G-automata, then $[\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$ iff $m(\mathscr{X}) \cong m(\mathscr{Y})$.*

Proof. We begin by assuming $[\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$. From Lemma 3.4.2.4 and Corollary 3.4.2.5 we know that $m(\mathscr{X})$ and $m(\mathscr{Y})$ are normal and accept the same language as $\mathscr{X}$ and $\mathscr{Y}$. From Lemma 3.4.2.3 it follows that $m(\mathscr{X})$ and $m(\mathscr{Y})$ are reachable. Proposition 3.4.2.6 thus implies that there exist G-automata homomorphisms $\pi_1 : m(\mathscr{Y}) \to m(\mathscr{X})$ and $\pi_2 : m(\mathscr{X}) \to m(\mathscr{Y})$. From the uniqueness property in Proposition 3.4.2.6 we deduce $\pi_1\pi_2 = \mathsf{id}_{m(\mathscr{X})}$ and $\pi_2\pi_1 = \mathsf{id}_{m(\mathscr{Y})}$. Hence $\pi_2 : m(\mathscr{X}) \to m(\mathscr{Y})$ is an isomorphism with inverse π_1.

Conversely, assume $m(\mathscr{X})$ is isomorphic to $m(\mathscr{Y})$. Then it immediately follows $[\![m(\mathscr{X})]\!] = [\![m(\mathscr{Y})]\!]$, which implies $[\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$ by Lemma 3.4.2.4. □

We conclude with the size-minimality of the minimisation of a normal automaton among language equivalent normal automata.

Corollary 3.4.2.10. *Let $\mathscr{X}$ and $\mathscr{Y}$ be normal G-automata with $[\![\mathscr{X}]\!] = [\![\mathscr{Y}]\!]$. Then $|m(\mathscr{X})| \leq |\mathscr{Y}|$, where $|m(\mathscr{X})| = |\mathscr{Y}|$ iff $m(\mathscr{X}) \cong \mathscr{Y}$.*

Proof. From Corollary 3.4.2.9 it immediately follows $m(\mathscr{X}) \cong m(\mathscr{Y})$. We additionally observe Figure 3.5a to derive

$$|m(\mathscr{X})| = |m(\mathscr{Y})| \leq |r(\mathscr{Y})| \leq |\mathscr{Y}|.$$

We show next $|m(\mathscr{X})| = |\mathscr{Y}|$ iff $m(\mathscr{X}) \cong \mathscr{Y}$:

- Assume $m(\mathscr{X}) \cong \mathscr{Y}$, then immediately $|m(\mathscr{X})| = |\mathscr{Y}|$.
- Assume $|m(\mathscr{X})| = |\mathscr{Y}|$, then Proposition 3.4.2.6 and Figure 3.5a imply:

$$|r(\mathscr{Y})| \geq |m(\mathscr{X})| = |\mathscr{Y}| \geq |r(\mathscr{Y})|.$$

It thus follows $|m(\mathscr{X})| = |r(\mathscr{Y})| = |\mathscr{Y}|$. From the second equality and the definition of $r(\mathscr{Y})$ it immediately follows $\mathscr{Y} \cong r(\mathscr{Y})$. The first equality implies that the epimorphism $\pi : r(\mathscr{Y}) \twoheadrightarrow m(\mathscr{X})$ in Proposition 3.4.2.6 is a bijective automata homomorphism. Any bijective coalgebra homomorphism is a coalgebra isomorphism [132, Prop. 2.3]. It is clear that the inverse of an initial state preserving coalgebra isomorphism preserves initial states as well. Thus $\pi : r(\mathscr{Y}) \cong m(\mathscr{X})$ is an G-automata isomorphism. □

3.5 Learning $m(\mathscr{X})$

In this section we formally investigate the correctness of GL^* (Algorithm 2). Our main result is Theorem 3.5.2.6, which shows that if the oracle is instantiated with a deterministic language accepted by a finite normal G-automaton $\mathscr{X}$, then GL^* terminates with a hypothesis isomorphic to $m(\mathscr{X})$. For calculations, it will be convenient to use the following definition of an observation table.

Definition 3.5.0.1. An *observation table* $T = (S, E, \mathsf{row})$ consists of subsets $S \subseteq \mathsf{GS}^-, E \subseteq \mathsf{GS}$ and a function $\mathsf{row} : S \cup S \cdot (\mathsf{At} \cdot \Sigma) \to 2^E$, such that:

- $\varepsilon \in S$ and $\mathsf{At} \subseteq E$
- $\alpha p e \in E$ implies $e \in E$ (suffix-closed)
- $s \alpha p \in S$ implies $s \in S$ (prefix-closed)
- $s \neq t$ implies $\mathsf{row}(s) \neq \mathsf{row}(t)$ for $s, t \in S$
- $\varepsilon \neq s \in S$ implies $\mathsf{row}(s)(e) = 1$ for some $e \in E$
- $\mathsf{row}(s \alpha p)(e) = \mathsf{row}(s)(\alpha p e)$, if $\alpha p e \in E$

Not every table induces a well-defined G-automaton. To ensure correctness, we have to restrict ourselves to a subclass of tables that satisfies two important properties. We call an observation table *deterministic* if the guarded string language $\mathsf{row}(s) \subseteq \mathsf{GS}$ is deterministic for all $s \in S$. An observation table is *closed*, if for all $t \in S \cdot (\mathsf{At} \cdot \Sigma)$ with $\mathsf{row}(t)(e) = 1$ for some $e \in E$, there exists an $s \in S$ such that $\mathsf{row}(s) = \mathsf{row}(t)$.

The result below shows that if the oracle is instantiated with a deterministic language accepted by a finite normal G-automaton $\mathscr{X}$, we have a well-defined observation table at every step.

Proposition 3.5.0.2. *If Algorithm 2 is instantiated with a deterministic language accepted by a finite normal G-automaton $\mathscr{X}$, then T is a well-defined deterministic observation table at every step.*

Proof.

- Any G-automaton accepts a deterministic language. Since $\mathsf{row}(s) \subseteq s^{-1}[\![\mathscr{X}]\!]$, and determinacy is preserved under derivatives, the determinacy of T is thus implied by the determinacy of $[\![\mathscr{X}]\!]$.

- In the initial step we have $S = \{\varepsilon\}$ and $E = \mathsf{At}$. In every step that follows the sets S and E are only extended. We thus have $\varepsilon \in S$ and $\mathsf{At} \subseteq E$ in every step.

- In the initial step $S = \{\varepsilon\}$ and $E = \mathsf{At}$ are clearly prefix and suffix closed, respectively. In the following steps S is only extended with strings of the shape $s\alpha p$ for $s \in S$, and E is only extended with the suffixes of some z. Hence S and E are prefix and suffix closed, respectively, in every step.

- In the initial step $S = \{\varepsilon\}$, hence all rows indexed by elements in S are trivially disjoint. In the following steps we only add $s\alpha p$ to S, if $\mathsf{row}(s\alpha p) \neq \mathsf{row}(t)$ for all $t \in S$. Since disjoint rows do at no point collapse, we can deduce that $s \neq t$ implies $\mathsf{row}(s) \neq \mathsf{row}(t)$ for all $s, t \in S$ in every step.

- In the initial step we have $S = \{\varepsilon\}$, thus the observation that for all $s \in S$ with $s \neq \varepsilon$ we have $\mathsf{row}(s) \neq \emptyset$ is trivially true. In the following steps we only add elements $s\alpha p$ with $\mathsf{row}(s\alpha p) \neq \emptyset$ to S.

- Since $\mathsf{row}(t)(e) = [\![\mathscr{X}]\!](te)$, the identity $\mathsf{row}(s\alpha p)(e) = \mathsf{row}(s)(\alpha pe)$, if $\alpha pe \in E$, follows from the associativity of string concatenation. □

Any closed deterministic table induces a G-automaton in the following way.

Definition 3.5.0.3. Given a closed deterministic observation table $T = (S, E, \mathsf{row})$, let $m(T) := (\{\mathsf{row}(s) \mid s \in S\}, \delta, \mathsf{row}(\varepsilon))$ be the G-automaton with

$$\delta(L)(\alpha) = \begin{cases} (p, (\alpha p)^{-1}L) & \text{if } (\alpha p)^{-1}L \neq \emptyset \\ 1 & \text{if } \alpha \in L \\ 0 & \text{otherwise} \end{cases}, \tag{3.7}$$

where $L \in \{\mathsf{row}(s) \mid s \in S\}$ and $(\alpha p)^{-1}\mathsf{row}(s) = \mathsf{row}(s\alpha p)$.

A few remarks on the well-definedness of the above definition are in order. By Definition 3.5.0.1 the upper-rows of an observation table are disjoint. Since T is deterministic, precisely one of the three cases in (3.7) occurs. If $(\alpha p)^{-1}\mathsf{row}(s)$ is non-empty, there exists, because T is closed, some $t \in S$ with $(\alpha p)^{-1}\mathsf{row}(s) = \mathsf{row}(t)$. This shows that $m(T)$ is closed under transitions.

3.5.1 Properties of $m(T)$

In what follows, let T be a closed deterministic observation table, unless states otherwise. We will establish a few basic properties of $m(T)$. First, we observe its reachability, which is implied by a slightly stronger statement.

Lemma 3.5.1.1. *For all $s \in S$ and $t \in \mathsf{GS}^-$ such that $st \in S$, we have $\mathsf{row}(s) \xrightarrow{t} \mathsf{row}(st)$ in $m(T)$. In particular, $m(T)$ is reachable.*

Proof. We show the statement by induction on the length of $t \in \mathsf{GS}^-$:

- If $t = \varepsilon$, the statement follows from the base case of (3.3), i.e. $\mathsf{row}(s) \xrightarrow{\varepsilon} \mathsf{row}(s)$.
- If $t = v\alpha p$ for $v \in \mathsf{GS}^-$, we have $sv \in S$, since $sv\alpha p = st \in S$ and S is prefix closed by Definition 3.5.0.1. Thus $\mathsf{row}(s) \xrightarrow{v} \mathsf{row}(sv)$ by the induction hypothesis. Since $\varepsilon \neq st \in S$, we have $(\alpha p)^{-1}\mathsf{row}(sv) = \mathsf{row}(st) \neq \emptyset$ by Definition 3.5.0.1. Thus it follows $\mathsf{row}(sv) \xrightarrow{\alpha p} \mathsf{row}(sv\alpha p)$ by (3.7). We conclude $\mathsf{row}(s) \xrightarrow{v\alpha p = t} \mathsf{row}(sv\alpha p) = \mathsf{row}(st)$ by (3.3).

Since $\epsilon \in S$ by Definition 3.5.0.1, we in particular obtain $\mathsf{row}(\varepsilon) \xrightarrow{s} \mathsf{row}(s)$ in $m(T)$ for all $s \in S$, which implies the reachability of $m(T)$. □

We call a G-automaton $(\mathscr{Y}, y)$ *consistent with* T, if $S \subseteq R(y)$ and $[\![y_s]\!](e) = \mathsf{row}(s)(e)$ for all $s \in S$, $e \in E$, and $y_s \in Y$ with $y \xrightarrow{s} y_s$. By Lemma 3.5.1.1, the automaton $m(T)$ is consistent with T if and only if $[\![\mathsf{row}(s)]\!](e) = \mathsf{row}(s)(e)$ for all $s \in S$ and $e \in E$. The consistency of $m(T)$ with T should not be confused with the consistency of T itself. Both terminologies appear frequently in the literature [14]. We show that $m(T)$ is not only consistent with T, but has in fact the fewest number of states among all automata consistent with T.

Lemma 3.5.1.2. *$m(T)$ is size-minimal among automata consistent with T.*

Proof. We begin by showing that $m(T)$ is consistent with T, that is, it satisfies $[\![\mathsf{row}(s)]\!](e) = \mathsf{row}(s)(e)$ for all $s \in S, e \in E$, by induction on the length of e:

- For the induction base, let $e = \alpha \in \mathsf{At}$, then it immediately follows that:

$$\begin{aligned} [\![\mathsf{row}(s)]\!](e) = 1 &\Leftrightarrow \delta(\mathsf{row}(s))(\alpha) = 1 && \text{(Definition of } [\![-]\!]) \\ &\Leftrightarrow \mathsf{row}(s)(\alpha) = 1 && (3.7) \end{aligned}$$

- For the induction step, let $e = \alpha p w$ for $w \in \mathsf{GS}$, then Definition 3.5.0.1 implies $w \in E$ and $\mathsf{row}(s)(\alpha p w) = \mathsf{row}(s\alpha p)(w)$. Thus we can deduce:

$$\begin{aligned} & [\![\mathsf{row}(s)]\!](\alpha p w) = 1 \\ \Leftrightarrow\ & \exists t \in S : \emptyset \neq \mathsf{row}(s\alpha p) = \mathsf{row}(t) \text{ and } [\![\mathsf{row}(t)]\!](w) = 1 && \text{(Definition of } [\![-]\!]) \\ \Leftrightarrow\ & \exists t \in S : \mathsf{row}(s\alpha p) = \mathsf{row}(t) \text{ and } \mathsf{row}(t)(w) = 1 && (w \in E, \text{ IH}) \\ \Leftrightarrow\ & \mathsf{row}(s\alpha p)(w) = 1 && (T \text{ closed}) \\ \Leftrightarrow\ & \mathsf{row}(s)(\alpha p w) = 1 && \text{(Definition 3.5.0.1)} \end{aligned}$$

Let $(\mathscr{Y}, y)$ be any G-automaton consistent with T, i.e. $S \subseteq R(y)$ and $[\![y_s]\!](e) = \mathsf{row}(s)(e)$ for all $s \in S, e \in E$, and $y_s \in Y$ with $y \xrightarrow{s} y_s$. We define a function $f : \{\mathsf{row}(s) \mid s \in S\} \to Y$ by $f(\mathsf{row}(s)) = y_s$. The function is well-defined, since

$S \subseteq R(y)$. Assume $f(\mathsf{row}(s)) = f(\mathsf{row}(t))$, i.e. $y_s = y_t$ for $s, t \in S$. Then we can deduce

$$\mathsf{row}(s)(e) = [\![y_s]\!](e) = [\![y_t]\!](e) = \mathsf{row}(t)(e)$$

for all $e \in E$. Since by Definition 3.5.0.1 rows indexed by S are disjoint, it follows $s = t$. This shows that f is injective, which implies the size-minimality of $m(T)$. □

From the consistency of $m(T)$ with T it is straightforward to derive its normality and observability.

Lemma 3.5.1.3. *$m(T)$ is normal and observable.*

Proof.

- Assume $\delta(\mathsf{row}(s))(\alpha) = (p, \mathsf{row}(t))$ for $s, t \in S$. Then we have

  $$\mathsf{row}(t) = \mathsf{row}(s\alpha p) = (\alpha p)^{-1}\mathsf{row}(s) \neq \emptyset$$

  by (3.7). From Lemma 3.5.1.2 $m(T)$ it follows that $[\![\mathsf{row}(t)]\!]$ is non-empty, which proves the normality of $m(T)$.

- Assume $[\![\mathsf{row}(s)]\!] = [\![\mathsf{row}(t)]\!]$ for $s, t \in S$. Then, by Lemma 3.5.1.2, we have

  $$\mathsf{row}(s)(e) = [\![\mathsf{row}(s)]\!](e) = [\![\mathsf{row}(t)]\!](e) = \mathsf{row}(t)(e)$$

  for all $e \in E \subseteq \mathsf{GS}$. Thus $\mathsf{row}(s) = \mathsf{row}(t)$, which shows that $[\![-]\!]$ is injective, that is, $m(T)$ is observable. □

3.5.2 Relationship Between $m(T)$ and $m(\mathscr{X})$

We will next deduce the correctness of GL^*, that is, its termination with an automaton isomorphic to $m(\mathscr{X})$, if the teacher is instantiated with the language accepted by a finite normal automaton $\mathscr{X}$.

In a first step we establish that any hypothesis admits an injective function from its state-space into the state-space of $m(\mathscr{X})$. The result below does not necessarily require the observation table to be deterministic or closed.

Lemma 3.5.2.1. *Let $T = (S, E, \mathsf{row})$ be an observation table with $\mathsf{row}(t)(e) = [\![\mathscr{X}]\!](te)$ for all $t \in S \cup S \cdot (\mathsf{At} \cdot \Sigma)$, $e \in E$, and let $x \in X$ be the initial state of*

$\mathscr{X}$. *Then* $\pi : \{\mathsf{row}(s) \mid s \in S\} \to \{w^{-1}[\![\mathscr{X}]\!] \mid w \in R(x)\}$, $\mathsf{row}(s) \mapsto s^{-1}[\![\mathscr{X}]\!]$ *is a well-defined injective function.*

Proof. We first show that π is well-defined. To this end, we need to establish that i) $S \subseteq R(x)$; and ii) if $\mathsf{row}(s) = \mathsf{row}(t)$ for $s, t \in S$, then $s^{-1}[\![\mathscr{X}]\!] = t^{-1}[\![\mathscr{X}]\!]$.

For i) note that if $s = \varepsilon$, then $x \xrightarrow{s} x$ by the base case of (3.3), i.e. $s \in R(x)$. If $s \neq \varepsilon$, then Definition 3.5.0.1 implies the existence of some $e \in E$, such that $\mathsf{row}(s)(e) = 1$. Thus $[\![x]\!](se) = [\![\mathscr{X}]\!](se) = \mathsf{row}(s)(e) = 1$, which implies $s \in R(x)$ by the definition of $[\![-]\!]$. For ii) it is enough to observe that by Definition 3.5.0.1 all rows of an observation table are disjoint.

To show that π is injective, assume $\pi(\mathsf{row}(s)) = \pi(\mathsf{row}(t))$, for $s, t \in S$. By definition of π we thus have an equivalence $s \equiv_{[\![\mathscr{X}]\!]} t$. From the definition of $\equiv_{[\![\mathscr{X}]\!]}$ and the assumptions it thus follows

$$e \in \mathsf{row}(s) \Leftrightarrow se \in [\![\mathscr{X}]\!] \Leftrightarrow te \in [\![\mathscr{X}]\!] \Leftrightarrow e \in \mathsf{row}(t)$$

for all $e \in E$. This proves the equality $\mathsf{row}(s) = \mathsf{row}(t)$. □

If the algorithm terminates with a hypothesis $m(T)$, the latter is, by definition, language equivalent to $\mathscr{X}$, and thus to the minimisation $m(\mathscr{X})$, by Lemma 3.4.2.4. The next result implies a stronger statement: in case of termination, the hypothesis $m(T)$ is *isomorphic* to $m(\mathscr{X})$, via the function π of Lemma 3.5.2.1.

Proposition 3.5.2.2. *Let* $T = (S, E, \mathsf{row})$ *be a closed deterministic observation table with* $\mathsf{row}(t)(e) = [\![\mathscr{X}]\!](te)$ *for all* $t \in S \cup S \cdot (\mathsf{At} \cdot \Sigma)$, $e \in E$. *Let* π *be the injection of Lemma 3.5.2.1, and* $\mathscr{X}$ *normal. The following are equivalent:*

1. $\pi : m(T) \simeq m(\mathscr{X})$ *is a G-automata isomorphism*

2. $[\![m(T)]\!] = [\![m(\mathscr{X})]\!]$

Proof.

- 1. → 2.: Since π is a homomorphism, it follows $[\![-]\!]_{m(\mathscr{X})} \circ \pi = [\![-]\!]_{m(T)}$

by uniqueness. In particular we have:

$$\begin{aligned}
[\![m(T)]\!]_{m(T)} &= [\![\mathsf{row}(\varepsilon)]\!]_{m(T)} && \text{(Definition of } [\![-]\!]_{m(T)}\text{)} \\
&= [\![\pi(\mathsf{row}(\varepsilon))]\!]_{m(\mathscr{X})} && ([\![-]\!]_{m(\mathscr{X})} \circ \pi = [\![-]\!]_{m(T)}) \\
&= [\![\varepsilon^{-1}\mathscr{X}]\!]_{m(\mathscr{X})} && \text{(Definition of } \pi\text{)} \\
&= [\![\mathscr{X}]\!]_{m(\mathscr{X})} && \text{(Definition of } [\![-]\!]_{m(\mathscr{X})}\text{)}
\end{aligned}$$

- 2. $\to$ 1. : By Lemma 3.5.1.3 and Lemma 3.5.1.1, $m(T)$ is normal and reachable. From the assumption and Lemma 3.4.2.4 it follows $[\![m(T)]\!] = [\![m(\mathscr{X})]\!] = [\![\mathscr{X}]\!]$. By assumption $\mathscr{X}$ is normal. By Proposition 3.4.2.6 there thus exists a unique surjective automata homomorphism $f\colon m(T) = r(m(T)) \to m(\mathscr{X})$ that satisfies $f(\mathsf{row}(s)) = w_s^{-1}[\![\mathscr{X}]\!]$ for $\mathsf{row}(\varepsilon) \xrightarrow{w_s} \mathsf{row}(s)$ in $m(T)$. From Lemma 3.5.1.1 it follows that $w_s = s$. Therefore the definitions of π and f coincide, $\pi = f$. Since by Lemma 3.5.2.1 the function $\pi = f$ is injective, it is a bijective coalgebra homomorphism. Any bijective coalgebra homomorphism is a coalgebra isomorphism [132, Prop. 2.3]. It is clear that the inverse of an initial state preserving coalgebra isomorphism preserves initial states as well. It thus follows that $\pi = f$ is a G-automata isomorphism. □

The main argument in the proof of Theorem 3.5.2.6 is Proposition 3.5.2.5. To prove the latter, we need the following two results. Both results assume two closed deterministic tables T and T', with the latter extending the former. The first statement, Lemma 3.5.2.3, relates the transition function of $m(T)$ to the one of $m(T')$.

Lemma 3.5.2.3. *Let $T = (S, E, \mathsf{row})$ and $T' = (S, E', \mathsf{row}')$ be closed deterministic observation tables with $E \subseteq E'$ and $\mathsf{row}(t)(e) = \mathsf{row}'(t)(e)$ for all $t \in S \cup S\cdot(\mathsf{At}\cdot\Sigma)$, $e \in E$. Let $m(T)$ and $m(T')$ have transition functions δ and δ', respectively, then for all $s, t \in S$:*

- $\delta'(\mathsf{row}'(s))(\alpha) = 1$ *iff* $\delta(\mathsf{row}(s))(\alpha) = 1$
- $\delta'(\mathsf{row}'(s))(\alpha) = (p, \mathsf{row}'(t))$ *implies* $\delta(\mathsf{row}(s))(\alpha) = (p, \mathsf{row}(t))$ *or* $\delta(\mathsf{row}(s))(\alpha) = 0$

- $\delta'(\mathsf{row}'(s))(\alpha) = 0$ *implies* $\delta(\mathsf{row}(s))(\alpha) = 0$

Proof. • For the first point we deduce:

$$\begin{aligned}
\delta'(\mathsf{row}'(s))(\alpha) = 1 &\Leftrightarrow \mathsf{row}'(s)(\alpha) = 1 && \text{(Definition of } \delta') \\
&\Leftrightarrow \mathsf{row}(s)(\alpha) = 1 && (\alpha \in \mathsf{At} \subseteq E) \\
&\Leftrightarrow \delta(\mathsf{row}(s))(\alpha) = 1 && \text{(Definition of } \delta)
\end{aligned}$$

- For the second point, assume $\delta'(\mathsf{row}'(s))(\alpha) = (p, \mathsf{row}'(t))$ for $t \in S$ with $\mathsf{row}'(s\alpha p) = \mathsf{row}'(t)$. Then by the first point

$$\delta(\mathsf{row}(s))(\alpha) = 0, \quad \text{or} \quad \delta(\mathsf{row}(s))(\alpha) = (p, \mathsf{row}(u))$$

  for some $u \in S$ with $\mathsf{row}(s\alpha p) = \mathsf{row}(u)$. We further have

$$\mathsf{row}(t)(e) = \mathsf{row}'(t)(e) = \mathsf{row}'(s\alpha p)(e) = \mathsf{row}(s\alpha p)(e) = \mathsf{row}(u)(e)$$

  for all $e \in E$. In other words, we have derived $\mathsf{row}(t) = \mathsf{row}(u)$.

- For the last point, assume $\delta'(\mathsf{row}'(s))(\alpha) = 0$. Then by the first point

$$\delta(\mathsf{row}(s))(\alpha) = 0, \quad \text{or} \quad \delta(\mathsf{row}(s))(\alpha) = (p, \mathsf{row}(t))$$

  for some $t \in S$ with $\mathsf{row}(s\alpha p) = \mathsf{row}(t)$. By the definition of δ, the latter case implies

$$\mathsf{row}'(s\alpha p)(e) = \mathsf{row}(s\alpha p)(e) = 1$$

  for some $e \in E \subseteq E'$. It thus follows $\delta'(row'(s))(\alpha) \notin 2$, which contradicts the assumption $\delta'(\mathsf{row}'(s))(\alpha) = 0$. We thus can conclude $\delta(\mathsf{row}(s))(\alpha) = 0$. □

The second statement, Lemma 3.5.2.4, establishes an inclusion of the language semantics of $m(T)$ into the language semantics of $m(T')$.

Lemma 3.5.2.4. *Let* $T = (S, E, \mathsf{row})$ *and* $T' = (S, E', \mathsf{row}')$ *be closed deterministic observation table with* $E \subseteq E'$ *and* $\mathsf{row}(t)(e) = \mathsf{row}'(t)(e)$ *for all* $t \in S \cup S \cdot (\mathsf{At} \cdot \Sigma)$, $e \in E$. *Then* $[\![\mathsf{row}(s)]\!]_{m(T)} \subseteq [\![\mathsf{row}'(s)]\!]_{m(T')}$ *for all* $s \in S$.

Proof. We show $w \in [\![\mathsf{row}(s)]\!]_{m(T)}$ implies $w \in [\![\mathsf{row}'(s)]\!]_{m(T')}$ for all $s \in S$ and $w \in \mathsf{GS}$ by induction on w. We denote the transition functions of $m(T)$ and $m(T')$ by δ and δ', respectively.

- For the induction base, assume $w = \alpha$. Then we deduce:

$$\begin{aligned}
\alpha \in [\![\mathsf{row}(s)]\!]_{m(T)} &\Leftrightarrow \delta(\mathsf{row}(s))(\alpha) = 1 && \text{(Definition of } [\![-]\!]) \\
&\Leftrightarrow \delta'(\mathsf{row}'(s))(\alpha) = 1 && \text{(Lemma 3.5.2.3)} \\
&\Leftrightarrow \alpha \in [\![\mathsf{row}'(s)]\!]_{m(T')} && \text{(Definition of } [\![-]\!])
\end{aligned}$$

- In the induction step, let $w = \alpha p v$. Then we have:

$$\begin{aligned}
& \alpha p v \in [\![\mathsf{row}(s)]\!]_{m(T)} \\
\Leftrightarrow\ & \text{(Definition of } [\![-]\!]) \\
& \exists t \in S : \delta(\mathsf{row}(s))(\alpha) = (p, \mathsf{row}(t)),\ v \in [\![\mathsf{row}(t)]\!]_{m(T)} \\
\Rightarrow\ & \text{(Lemma 3.5.2.3, IH)} \\
& \exists t \in S : \delta'(\mathsf{row}'(s))(\alpha) = (p, \mathsf{row}'(t)),\ v \in [\![\mathsf{row}'(t)]\!]_{m(T')} \\
\Leftrightarrow\ & \text{(Definition of } [\![-]\!]) \\
& \alpha p v \in [\![\mathsf{row}'(s)]\!]_{m(T')}
\end{aligned}$$

□

The next result shows that, if the oracle replies *no* to an equivalence query and provides us with a counterexample z, then the table extended with the suffixes of z can immediately be closed only if it is the first time such a situation occurs.

Proposition 3.5.2.5. *Let $T = (S, E, \mathsf{row})$ be a closed deterministic observation table with $\mathsf{row}(t)(e) = [\![\mathscr{X}]\!](te)$ for all $t \in S \cup S \cdot (\mathsf{At} \cdot \Sigma)$, $e \in E$. Let $[\![m(T)]\!](z) \neq [\![\mathscr{X}]\!](z)$ for some $z \in \mathsf{GS}$, and $T' = (S, E \cup \mathsf{suf}(z), \mathsf{row}')$ with $\mathsf{row}'(t)(e) = [\![\mathscr{X}]\!](te)$. If T' is closed, then $\mathsf{row}'(\varepsilon)(e) = 0$ for all $e \in E$, but $\mathsf{row}'(\varepsilon)(z') = 1$ for some $z' \in \mathsf{suf}(z)$.*

Proof.

- Assume $[\![\mathscr{X}]\!](z) = 0$ and $[\![m(T)]\!](z) = 1$. Since $[\![m(T)]\!] \subseteq [\![m(T')]\!]$ by Lemma 3.5.2.4, it follows $[\![m(T')]\!](z) = 1$. From Lemma 3.5.1.2 and the global

assumptions we thus can deduce

$$1 = [\![m(T')]\!](z) = [\![\mathsf{row}'(\varepsilon)]\!](z) = \mathsf{row}'(\varepsilon)(z) = [\![\mathscr{X}]\!](z),$$

which contradicts $0 = [\![\mathscr{X}]\!](z)$.

- Assume $[\![\mathscr{X}]\!](z) = 1$ and $[\![m(T)]\!](z) = 0$. From Lemma 3.5.1.2 and the global assumptions we can deduce

  $$1 = [\![\mathscr{X}]\!](z) = \mathsf{row}'(\varepsilon)(z) = [\![\mathsf{row}'(\varepsilon)]\!](z) = [\![m(T')]\!](z).$$

  By Lemma 3.5.2.3, there exists some decomposition $z = v\alpha p z'$ for $v \in \mathsf{GS}^-, z' \in \mathsf{suf}(z)$, such that for some $t \in S$:

  1. $\mathsf{row}'(\varepsilon) \xrightarrow{v} \mathsf{row}'(t)$ in $m(T')$ and $\mathsf{row}(\varepsilon) \xrightarrow{v} \mathsf{row}(t)$ in $m(T)$;
  2. $\delta'(\mathsf{row}'(t))(\alpha) = (p, \mathsf{row}'(t'))$ for some $t' \in S$, but $\delta(\mathsf{row}(t))(\alpha) = 0$.

  For all $e \in E$ it follows

  $$0 = \mathsf{row}(t\alpha p)(e) = \mathsf{row}'(t\alpha p)(e) = \mathsf{row}'(t')(e) = \mathsf{row}(t')(e),$$

  since otherwise $\delta(\mathsf{row}(t))(\alpha) \neq 0$ by the definition of δ. From Definition 3.5.0.1 we can deduce $t' = \varepsilon$. Since $(v\alpha p)z' = z \in [\![m(T')]\!] = [\![\mathsf{row}'(\varepsilon)]\!]$ and $\mathsf{row}'(\varepsilon) \xrightarrow{v\alpha p} \mathsf{row}'(t') = \mathsf{row}'(\varepsilon)$, it follows $z' \in [\![\mathsf{row}'(\varepsilon)]\!]$ by the definition of $[\![-]\!]$. From Lemma 3.5.1.2 we can conclude $z' \in \mathsf{row}'(\varepsilon)$. □

In consequence, an infinite chain of negative equivalence queries and immediately closed extended tables is impossible. Since fixing a closedness defect increases the size of $m(T)$, which by Lemma 3.5.2.1 is bounded by the finite number of states in $m(\mathscr{X})$, we can deduce the correctness of Algorithm 2.

Theorem 3.5.2.6. *If Algorithm 2 is instantiated with the language accepted by a finite normal automaton $\mathscr{X}$, then it terminates with a hypothesis isomorphic to $m(\mathscr{X})$.*

Proof. By Proposition 3.5.0.2, T is a well-defined deterministic observation table at every step. We continue by showing that the algorithm yields $m(\mathscr{X})$ in finitely

many steps. By Lemma 3.5.2.1 we have $|X| \leq |Y|$ for $X = \{\mathsf{row}(s) \mid s \in S\}$ and $Y = \{w^{-1}[\![\mathscr{X}]\!] \mid w \in R(x)\}$ at any point of the algorithm. Since $\mathscr{X}$ is finite, the state-space Y of $m(\mathscr{X})$ is finite. At no point of the algorithm does the size of X decrease. Resolving a closedness defect strictly increases the size of X. Hence a closedness defect can only occur finitely many times. The only way the algorithm could not terminate is thus an infinite chain of negative equivalence queries, for which the subsequent suffix-enriched table is immediately closed again. By applying Proposition 3.5.2.5 twice, one observes that such a case can not occur. □

3.6 Comparison with Moore Automata

How are the minimal GKAT automaton (Figure 3.2e) and the minimal Moore automaton (Figure 3.1f) representing the guarded deterministic language (3.1) related? Why should we learn the former, and not the latter? Are there optimizations for L^* that we could adapt for GL^*? Those are the questions this section seeks to answer.

3.6.1 Embedding of GKAT Automata

Comparing the GKAT automaton in Figure 3.2e with the Moore automaton (with input alphabet $\mathsf{At} \cdot \Sigma$ and output alphabet 2^{At}, short *M-automaton*) in Figure 3.1f suggests that the latter can be recovered from the former by adding a sink-state to make halting transitions explicit. Lemma 3.6.1.1 below formalises this idea.

For completeness, we first briefly recall the language semantics of Moore automata. Let $\mathscr{X} = (X, \langle \varepsilon, \delta \rangle)$ be a M-coalgebra, where $MX = B \times X^A$, for an input alphabet A and an output alphabet B. Then one can inductively define a function $[\![-]\!] : X \to B^{A^*}$ via $[\![x]\!](\varepsilon) = \varepsilon(x)$ and $[\![x]\!](av) = [\![\delta(x)(a)]\!](v)$. In particular, if $A = (\mathsf{At} \cdot \Sigma)$ and $B = 2^{\mathsf{At}}$ for finite sets At and Σ, then the former induces via currying a semantics function $[\![-]\!] : X \to \mathcal{P}((\mathsf{At} \cdot \Sigma)^* \cdot \mathsf{At}) = \mathcal{P}(\mathsf{GS})$ that is defined by:

$$\alpha \in [\![x]\!] \Leftrightarrow \varepsilon(x)(\alpha) = 1; \qquad \alpha p v \in [\![x]\!] \Leftrightarrow \delta(x)(\alpha p) = y \text{ and } v \in [\![y]\!].$$

Lemma 3.6.1.1. *Given a G-automaton* $\mathscr{X} = (X, \delta, x)$*, let* $f(\mathscr{X}) := (X + \{\star\}, \langle \varepsilon, \partial \rangle, x)$

be the M-automaton with:

$$\partial(x)(\alpha p) := \begin{cases} y & \text{if } x \in X,\ \delta(x)(\alpha) = (p, y) \\ \star & \text{otherwise} \end{cases}$$

$$\varepsilon(x)(\alpha) := \begin{cases} 1 & \text{if } x \in X,\ \delta(x)(\alpha) = 1 \\ 0 & \text{otherwise} \end{cases}.$$

Then $[\![x]\!]_{\mathscr{X}} = [\![x]\!]_{f(\mathscr{X})}$ for all $x \in X$, and $[\![\star]\!]_{f(\mathscr{X})} = \emptyset$. In particular, $[\![f(\mathscr{X})]\!]_{f(\mathscr{X})} = [\![\mathscr{X}]\!]_{\mathscr{X}}$.

Proof. We simultaneously show i) $w \notin [\![\star]\!]_{f(\mathscr{X})}$ and ii) $w \in [\![x]\!]_{f(\mathscr{X})}$ iff $w \in [\![x]\!]_{\mathscr{X}}$, for all $w \in \mathsf{GS}$ and $x \in X$ by induction on the length of w.

- For the induction base assume $w = \varepsilon$, then we find for i):

$$\begin{aligned} \alpha \in [\![\star]\!]_{f(\mathscr{X})} &\Leftrightarrow \varepsilon(\star)(\alpha) = 1 && \text{(Definition of } [\![-]\!]_{f(\mathscr{X})}) \\ &\Leftrightarrow 0 = 1 && \text{(Definition of } \varepsilon) \\ &\Leftrightarrow \text{false} && (0 \neq 1) \end{aligned}$$

  Similarly, we derive the following chain for ii):

$$\begin{aligned} \alpha \in [\![x]\!]_{f(\mathscr{X})} &\Leftrightarrow \varepsilon(x)(\alpha) = 1 && \text{(Definition of } [\![-]\!]_{f(\mathscr{X})}) \\ &\Leftrightarrow \delta(x)(\alpha) = 1 && \text{(Definition of } \varepsilon,\ x \in X) \\ &\Leftrightarrow \alpha \in [\![x]\!]_{\mathscr{X}} && \text{(Definition of } [\![-]\!]_{\mathscr{X}}) \end{aligned}$$

- For the induction step, let $w = \alpha p v$ for $v \in \mathsf{GS}$, then we deduce for i):

$$\begin{aligned} \alpha p v \in [\![\star]\!]_{f(\mathscr{X})} &\Leftrightarrow \partial(\star)(\alpha p) = y,\ v \in [\![y]\!]_{f(\mathscr{X})} && \text{(Definition of } [\![-]\!]_{f(\mathscr{X})}) \\ &\Leftrightarrow \partial(\star)(\alpha p) = \star,\ v \in [\![\star]\!]_{f(\mathscr{X})} && \text{(Definition of } \partial) \\ &\Leftrightarrow \text{false} && \text{(IH)} \end{aligned}$$

Analogously, we deduce for ii):

$$
\begin{aligned}
& \alpha p v \in [\![x]\!]_{f(\mathscr{X})} \\
\Leftrightarrow\ & \text{(Definition of } [\![-]\!]_{f(\mathscr{X})}\text{)} \\
& \partial(x)(\alpha p) = y,\ v \in [\![y]\!]_{f(\mathscr{X})} \\
\Leftrightarrow\ & \text{(Definition of } \partial\text{)} \\
& \delta(x)(\alpha) = (p, y),\ v \in [\![y]\!]_{f(\mathscr{X})} \text{ or } \delta(x)(\alpha) \in 2,\ v \in [\![\star]\!]_{f(\mathscr{X})} \\
\Leftrightarrow\ & \text{(IH)} \\
& \delta(x)(\alpha) = (p, y),\ v \in [\![y]\!]_{\mathscr{X}} \text{ or false} \\
\Leftrightarrow\ & \text{(Definition of } [\![-]\!]_{\mathscr{X}}\text{)} \\
& \alpha p v \in [\![x]\!]_{\mathscr{X}}
\end{aligned}
$$

In particular, $[\![f(\mathscr{X})]\!]_{f(\mathscr{X})} = [\![x]\!]_{f(\mathscr{X})} = [\![x]\!]_{\mathscr{X}} = [\![\mathscr{X}]\!]_{\mathscr{X}}$. □

As one would hope for, the above construction maps, up to isomorphism, the minimal GKAT automaton $m(\mathscr{X})$ to the minimal Moore automaton accepting the same language as $\mathscr{X}$.

Corollary 3.6.1.2. *Let $\mathscr{X}$ be a normal G-automaton, then $f(m(\mathscr{X})) \cong m(f(\mathscr{X}))$ as M-automata.*

Proof. From Lemma 3.6.1.1 and Lemma 3.4.2.4 we can deduce that $f(m(\mathscr{X}))$ accepts $[\![\mathscr{X}]\!]$, which is also accepted by $m(f(\mathscr{X}))$.

By Corollary 3.4.2.5 $[\![-]\!]_{m(\mathscr{X})}$ is injective. As Corollary 3.4.2.5 and Lemma 3.4.2.3 imply that $m(\mathscr{X})$ is normal and reachable, we know that $[\![-]\!]_{m(\mathscr{X})}$ never evaluates to the empty set. From Lemma 3.6.1.1 we thus can deduce that $[\![-]\!]_{f(m(\mathscr{X}))}$ is injective.

It is not hard to see that if a state is reachable in $\mathscr{Y}$, then it is reachable in $f(\mathscr{Y})$. The element $\star$ is reachable in $f(\mathscr{Y})$ in particular if $\mathscr{Y}$ is reachable and normal. Hence, since $m(\mathscr{X})$ is reachable and normal by Lemma 3.4.2.3 and Corollary 3.4.2.5, respectively, $f(m(\mathscr{X}))$ is reachable.

The automaton $f(m(\mathscr{X}))$ thus accepts the same language as $m(f(\mathscr{X}))$, is observable, and reachable. By uniqueness, $f(m(\mathscr{X}))$ and $m(f(\mathscr{X}))$ are thus isomorphic. □

3.6.2 Complexity Analysis

We now compare the worst-case complexities of L^* (Algorithm 1) and GL^* (Algorithm 2) for learning automata representations of GKAT programs e. We are mainly interested in a bound to the number of membership queries to $[\![e]\!]$. The example runs in Figure 3.1 and Figure 3.2 seem to indicate that with respect to this aspect, GL^* performs better than L^*. The result below confirms this intuition.

Proposition 3.6.2.1. *Algorithm 1 requires at most $O(a*(|\mathsf{At}|*b))$ many membership queries to $[\![e]\!]$ for learning a M-automaton representation of e, whereas Algorithm 2 requires at most $O(a * (|\mathsf{At}| + b))$ many membership queries to $[\![e]\!]$ for learning a G-automaton representation of e, for some[9] integers $a, b \in \mathbb{N}$.*

Proof. We first derive the maximum number of entries a table indexed by S and E can have during a run of L^* for generalised languages with input alphabet A and output alphabet B. Since we use a suffix-strategy for the handling of counterexamples (opposed to a prefix-strategy), our presentation slightly differs from the one in [14]. Let k denote the cardinality of the alphabet A. The number of states of the minimal Moore automaton accepting the target language is referred to by n, and the maximum length of a counterexample by m. The size of S is bounded by n. In the worst case, the sets S and $S \cdot A$ are disjoint. The cardinality of $S \cup S \cdot A$ is thus bounded by $n + n * k$. The maximum number of strings in E is $1 + m * (n - 1)$. This is because E is instantiated with ε, and only extended with suffixes of counterexamples. Each counterexample has at most m suffixes, and there can only be $n-1$ counterexamples, since any counterexample leads to a closedness defect, resolving which increases the size of S, which is instantiated with ε, and bounded in size by n. A table can thus have at most $(n + n * k) * (1 + m * (n - 1))$, or $O(m * n^2 * k)$, many entries.

In the case of $A = (\mathsf{At} \cdot \Sigma)$ and $B = 2^{\mathsf{At}}$, each entry requires $|\mathsf{At}|$ many membership queries. Overall Algorithm 1 thus requires at most

$$O(n * |\mathsf{At}| * |\Sigma| * (|\mathsf{At}| * m * n))$$

many membership queries to learn a deterministic guarded string language.

[9] Let m be the maximum length of a counterexample and n the size of the minimal Moore automaton accepting $[\![e]\!]$, then $a = n * |\mathsf{At}| * |\Sigma|$ and $b = m * n$.

We now derive a bound to the number of membership queries GL^* requires. As before, let m denote the maximum length of a counterexample, and n the number of states of the minimal Moore automaton accepting the target language. The cardinality of the set S is bounded by the number of states in the minimal GKAT automaton accepting the target language, which by Corollary 3.6.1.2 is $n-1$. The cardinality of $S \cup S \cdot (\mathsf{At} \cdot \Sigma)$ is thus bounded by $(n-1) + (n-1) * |\mathsf{At}| * |\Sigma|$. The maximum number of strings in E is $|\mathsf{At}| + m * (n-1)$. This is because E is instantiated with At, and only extended with suffixes of counterexamples. Each counterexample has at most m suffixes, and there can only be $n-1$ counterexamples. Indeed, assume there are $u > n-1$ counterexamples. Since resolving a closedness defect increases the size of S, which is instantiated with ε, and in size bounded by $n-1$, there can be at most $(n-1)-1 = n-2$ counterexamples for which the extended table is not closed. Thus there must be at least $u-(n-2) = u-n+2 \geq n-n+2 = 2$ counterexamples for which the extended table is closed. This is a contradiction, since Proposition 3.5.2.5 implies that this can be the case for at most 1 counterexample. A table can thus have at most $((n-1)+(n-1)*|\mathsf{At}|*|\Sigma|)*(|\mathsf{At}|+m*(n-1))$ entries. Each entry requires one membership query. Overall Algorithm 2 requires at most

$$O(n * |\mathsf{At}| * |\Sigma| * (|\mathsf{At}| + m * n))$$

many membership queries to learn a deterministic guarded string language. The statement follows by setting $a := n * |\mathsf{At}| * |\Sigma|$ and $b := m * n$. □

The following result shows that for all integers x, y greater than 2, the product $x * y$ is strictly greater than the sum $x + y$.

Lemma 3.6.2.2. *Let $a, b \in \mathbb{N}_{\geq 3}$, then $a * b > a + b$.*

Proof. We prove the statement by induction on $a \in \mathbb{N}_{\geq 3}$.

- For the induction base, assume $a = 3$. We show that $3 * b > 3 + b$ for all $b \in \mathbb{N}_{\geq 3}$ by induction on b. In the induction base, $b = 3$, the statement immediately is implied by $3 * 3 = 9 > 6 = 3 + 3$. Assume the statement is true for some $b \geq 3$.

The induction step follows from

$$3 * (b + 1) = 3b + 3 > (3 + b) + 3 = 6 + b > 3 + (b + 1).$$

- Assume the statement is true for $a \geq 3$. The induction step follows from

$$(a + 1) * b = (a * b) + b > (a + b) + b > (a + 1) + b.$$ □

In fact, it is also not hard to see that the difference between $a*b$ and $a+b$ increases with the sizes of a and b.

Lemma 3.6.2.3. *$a + b$ lies in $o(a * b)$*

Proof. We need to show that for any positive real number $c > 0$, there exists a natural number N, such that for all natural numbers $a, b \geq N$, we have $a + b < c * (a * b)$, or equivalently, $\frac{a+b}{a*b} < c$.

Given $c > 0$, we define $N := \lceil \frac{2}{c} \rceil + 1$. Let $a, b \geq N > \frac{2}{c}$. Then it immediately follows that $\frac{c}{2} > \frac{1}{b}$ and $\frac{c}{2} > \frac{1}{a}$. Thus we can compute

$$\frac{a + b}{a * b} = \frac{a}{a * b} + \frac{b}{a * b} = \frac{1}{b} + \frac{1}{a} < \frac{c}{2} + \frac{c}{2} = c.$$

□

The advantage of GL^* over L^* for learning deterministic guarded string languages in terms of membership queries thus increases with the number of atoms, which is exponential in the number of primitive tests, $\mathsf{At} \cong 2^T$. In applications to network verification, the number of tests, thus atoms, is typically quite large [12]. The difference between GL^* and L^* described in Proposition 3.6.2.1 is mainly due to a subtle play with the table indices, based on currying. It can be further increased by avoiding querying certain rows all together, taking into account the deterministic nature of the target language, as indicated in Section 3.2.2.

3.6.3 Optimized Counterexamples

In this section we present an optimization of GL^* that is based on a subtle refinement of Proposition 3.5.2.5. We show that while Algorithm 2 reacts to a negative equivalence query with counterexample $z \in \mathsf{GS}$ by adding columns for *all* suffixes in $\mathsf{suf}(z)$, it is in fact enough to add columns for a smaller subset of suffixes $\mathsf{suf}(z') \subseteq \mathsf{suf}(z)$, for some $z' \in \mathsf{suf}(z)$ of minimal length. Our approach is inspired by the optimized counterexample handling method of Rivest and Schapire for L^* [130]. To state Lemma 3.6.3.1, we need to define the following set:

$$A_z := \{z' \in \mathsf{suf}(z) \mid \; z = v\alpha pz',\ \mathsf{row}(\varepsilon) \xrightarrow{v} \mathsf{row}(s_v),\ x \xrightarrow{s_v} x_{s_v},\\ [\![\mathsf{row}(s_v)]\!](\alpha pz') \neq [\![x_{s_v}]\!](\alpha pz')\}$$

Intuitively, A_z contains all strictly-shorter suffixes of z that witness a mismatch between the behaviour of the hypothesis and the target language.

Lemma 3.6.3.1. *Let $T = (S, E, \mathsf{row})$ be a closed deterministic observation table with $\mathsf{row}(t)(e) = [\![\mathscr{X}]\!](te)$ for all $t \in S \cup S \cdot (\mathsf{At} \cdot \Sigma)$, $e \in E$. Let $[\![m(T)]\!](z) \neq [\![\mathscr{X}]\!](z)$ for some $z \in \mathsf{GS}$, and $z' := \mathsf{min}(A_z)$. If $T' = (S, E \cup \mathsf{suf}(z'), \mathsf{row}')$ with $\mathsf{row}'(t)(e) = [\![\mathscr{X}]\!](te)$ is closed, then $\mathsf{row}'(\varepsilon)(e) = 0$ for all $e \in E$, but $\mathsf{row}'(\varepsilon)(z') = 1$.*

Proof. We begin by showing that $z \notin \mathsf{At}$. Let us assume the opposite, $z = \alpha \in \mathsf{At} \subseteq E$. In that case, it follows:

$$\begin{aligned}
[\![m(T)]\!](z) &= [\![\mathsf{row}(\varepsilon)]\!](\alpha) && \text{(Definition of } [\![-]\!],\ z) \\
&= \mathsf{row}(\varepsilon)(\alpha) && \text{(Lemma 3.5.1.2)} \\
&= [\![\mathscr{X}]\!](\alpha) && (\mathsf{row}(t)(e) = [\![\mathscr{X}]\!](te)) \\
&= [\![\mathscr{X}]\!](z) && \text{(Definition of } z)
\end{aligned}$$

which is a contradiction. Thus there exists a decomposition $z = \varepsilon\alpha pz'$ for some $z' \in \mathsf{suf}(z)$. The former immediately implies that the set A_z is non-empty. Hence the shortest suffix $z' := \mathsf{min}(A_z)$ is well-defined. By construction, we have $\mathsf{row}(\varepsilon) \xrightarrow{v} \mathsf{row}(s_v)$, $x \xrightarrow{s_v} x_{s_v}$, and $[\![\mathsf{row}(s_v)]\!](\alpha pz') \neq [\![x_{s_v}]\!](\alpha pz')$.

- Assume $[\![x_{s_v}]\!](\alpha pz') = 0$ and $[\![\mathsf{row}(s_v)]\!](\alpha pz') = 1$, then there exists $s_{v\alpha p} \in S$

with $\mathsf{row}(\varepsilon) \xrightarrow{v} \mathsf{row}(s_v) \xrightarrow{\alpha p} \mathsf{row}(s_{v\alpha p})$ such that:

$$\begin{aligned}
1 &= [\![\mathsf{row}(s_v)]\!](\alpha p z') && \text{(Assumption)}\\
&= [\![\mathsf{row}(s_{v\alpha p})]\!](z') && \text{(Definition of } [\![-]\!])\\
&= [\![\mathsf{row}'(s_{v\alpha p})]\!](z') && \text{(Lemma 3.5.2.4)}\\
&= \mathsf{row}'(s_{v\alpha p})(z') && \text{(Lemma 3.5.1.2)}\\
&= \mathsf{row}'(s_v\alpha p)(z') && \text{(Definition of } \mathsf{row}')\\
&= [\![\mathscr{X}]\!](s_v\alpha p z') && (\mathsf{row}'(t)(e) = [\![\mathscr{X}]\!](te))\\
&= [\![x_{s_v}]\!](\alpha p z') && \text{(Definition of } [\![-]\!])
\end{aligned}$$

which contradicts $0 = [\![x_{s_v}]\!](\alpha p z')$.

- Assume $[\![x_{s_v}]\!](\alpha p z') = 1$ and $[\![\mathsf{row}(s_v)]\!](\alpha p z') = 0$, then there exists $s_{v\alpha p} \in S$ with $\mathsf{row}'(\varepsilon) \xrightarrow{v} \mathsf{row}'(s_v) \xrightarrow{\alpha p} \mathsf{row}'(s_{v\alpha p})$ such that:

$$\begin{aligned}
1 &= [\![x_{s_v}]\!](\alpha p z') && \text{(Assumption)}\\
&= [\![\mathscr{X}]\!](s_v\alpha p z') && \text{(Definition of } [\![-]\!])\\
&= \mathsf{row}'(s_v\alpha p)(z') && (\mathsf{row}'(t)(e) = [\![\mathscr{X}]\!](te))\\
&= \mathsf{row}'(s_{v\alpha p})(z') && \text{(Definition of } \mathsf{row}')\\
&= [\![\mathsf{row}'(s_{v\alpha p})]\!](z') && \text{(Lemma 3.5.1.2)}\\
&= [\![\mathsf{row}'(s_v)]\!](\alpha p z') && \text{(Definition of } [\![-]\!])
\end{aligned}$$

By Lemma 3.5.2.3 there thus are two possibilities:

  - Assume $\mathsf{row}(s_v) \xrightarrow{\alpha p} \mathsf{row}(s_{v\alpha p})$, then there are two options:

    * If $z' \notin \mathsf{At}$, we find a contradiction to the minimality of z' in A_z.

∗ If $z' \in \mathsf{At} \subseteq E$, then:

$$\begin{aligned}
0 &= [\![\mathsf{row}(s_v)]\!](\alpha p z') && \text{(Assumption)}\\
&= [\![\mathsf{row}(s_{v\alpha p})]\!](z') && (\mathsf{row}(s_v) \xrightarrow{\alpha p} \mathsf{row}(s_{v\alpha p}))\\
&= \mathsf{row}(s_{v\alpha p})(z') && \text{(Lemma 3.5.1.2)}\\
&= [\![\mathscr{X}]\!](s_{v\alpha p} z') && (\mathsf{row}(t)(e) = [\![\mathscr{X}]\!](te))\\
&= [\![x_{s_{v\alpha p}}]\!](z') && \text{(Definition of } [\![-]\!])\\
&= [\![x_{s_v}]\!](\alpha p z') && \text{(Definition of } [\![-]\!])
\end{aligned}$$

which is a contradiction to $[\![x_{s_v}]\!](\alpha p z') = 1$.

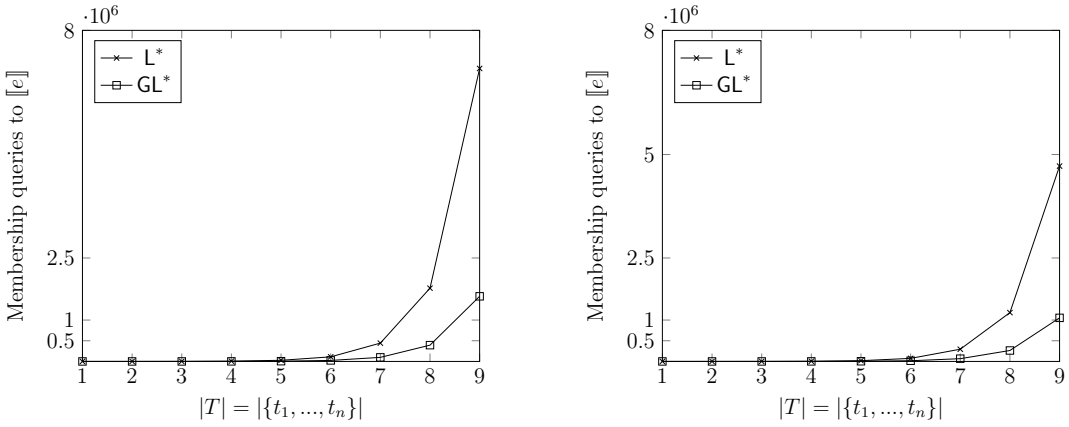
– Assume $\delta(\mathsf{row}(s_v))(\alpha) = 0$, then we have, for all $e \in E$:

$$\begin{aligned}
0 &= \mathsf{row}(s_v \alpha p)(e) && (\delta(\mathsf{row}(s_v))(\alpha) = 0)\\
&= \mathsf{row}'(s_v \alpha p)(e) && (\mathsf{row}(t)(e) = \mathsf{row}'(t)(e))\\
&= \mathsf{row}'(s_{v\alpha p})(e) && \text{(Definition of } \mathsf{row}')\\
&= \mathsf{row}(s_{v\alpha p})(e) && (\mathsf{row}(t)(e) = \mathsf{row}'(t)(e))
\end{aligned}$$

From Definition 3.5.0.1 it follows $s_{v\alpha p} = \varepsilon$, which implies the claim. □

Let z_0 be the shortest suffix of z and z_i the suffix of z of length $|z_{i-1}| + 1$. The suffix $\mathsf{min}(A_z)$ can be computed in at most $|\mathsf{suf}(z)| - 1$ steps: verify whether $z_i \in A_z$, beginning with z_0; if positive, break and set $\mathsf{min}(A_z) := z_i$, otherwise loop with z_{i+1}.

For example, if T is the closed table in Figure 3.2b with the corresponding hypothesis $m(T)$ in Figure 3.2c and counterexample $z = bp\overline{b}qb$, then $z' = \mathsf{min}(A_z) = \overline{b}qb$, since $b \notin A_z$. Lemma 3.6.3.1 shows that, instead of adding columns for the two non-present suffixes $bp\overline{b}qb$ and $\overline{b}qb$ of z, it is sufficient to add only one column for the single non-present suffix $\overline{b}qb$ of z'. In this case, the counterexample z is relatively short, thus the number of avoided columns small; in general, however, the advantage can be more significant.



(a) $e =$ `if` t_1 `then do` p_1 `else do` p_2 (b) $e = (\texttt{while}\ t_1\ \texttt{do}\ p_1); \texttt{do}\ p_2$

Figure 3.6: A comparison between GL^* and L^* with respect to membership queries

3.7 Implementation

We have implemented both GL^* and L^* in OCaml [122]; the code is available on GitHub[10]. The implementation allows one to compare, for any GKAT expression $e \in \mathsf{Expr}_{\Sigma,T}$, the number of membership queries to $[\![e]\!]$ required by GL^* for learning a G-automaton representation of e, with the number of membership queries to $[\![e]\!]$ required by L^* for learning a M-automaton representation of e. For each run, we output, for both algorithms, a trace of the involved hypotheses as tables in the `.csv` format and graphs in the `.dot` format, as well as an overview of the involved queries in the `.csv` format.

In Figure 3.6a we plot the results for the expression `if` t_1 `then do` p_1 `else do` p_2, the primitive actions $\Sigma = \{p_1, p_2, p_3\}$, and primitive tests $T = \{t_1, ..., t_n\}$ parametric in $n = 1, ..., 9$. We find that GL^* outperforms L^* for all choices of n. The difference in the number of membership queries increases with the size of n, as suggested by Proposition 3.6.2.1. For $n = 9$ the number of atoms is 2^9, resulting in an already relatively large number of queries for both algorithms. The picture is similar in Figure 3.6b, where we choose the expression $(\texttt{while}\ t_1\ \texttt{do}\ p_1); \texttt{do}\ p_2$, the primitive actions $\Sigma = \{p_1, p_2\}$, and primitive tests $T = \{t_1, ..., t_n\}$ parametric in $n = 1, ..., 9$.

[10] `https://github.com/zetzschest/gkat-automata-learning`

$\lvert\Sigma\rvert$	$\lvert T\rvert$	GL^*	L^*
3	1	26	114
3	2	100	444
3	3	392	1.752
3	4	1.552	6.960
3	5	6.176	27.744
3	6	24.640	110.784
3	7	98.432	442.752
3	8	393.472	1.770.240
3	9	1.573.376	7.079.424

(a) $e = \texttt{if } t_1 \texttt{ then do } p_1 \texttt{ else do } p_2$

$\lvert\Sigma\rvert$	$\lvert T\rvert$	GL^*	L^*
2	1	36	78
2	2	102	300
2	3	330	1.176
2	4	1.170	4.656
2	5	4.386	18.528
2	6	16.962	73.920
2	7	66.690	295.296
2	8	264.450	1.180.416
2	9	1.053.186	4.720.128

(b) $e = (\texttt{while } t_1 \texttt{ do } p_1)\texttt{; do } p_2$

Figure 3.7: The exact number of membership queries to $[\![e]\!]$ underlying the comparison between GL^* and L^* in Figure 3.6

Again, GL^* requires significantly less queries in all cases of n, and the difference increases with the size of n. The exact numbers of membership queries underlying the graphs in Figure 3.6 can be found in Figure 3.7.

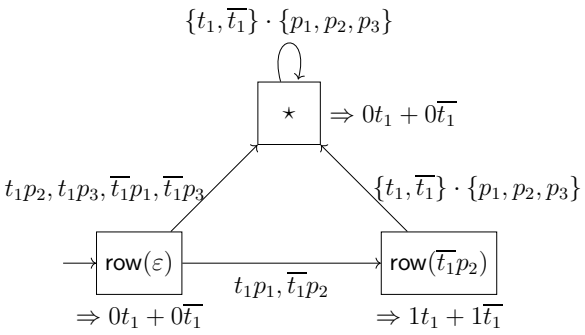
In Figure 3.8 and Figure 3.9 we give concrete examples of the tables and corresponding automata our implementations of GL^* and L^* deduce. It is instructive to recall Corollary 3.6.1.2 in this context: the embedding in Figure 3.8b is clearly isomorphic to the minimal Moore automaton in Figure 3.9b.

Our implementation generates an oracle for L^* from a GKAT expression e in the following way. First, we interpret e as a KAT expression $\iota(e)$ via the standard embedding of GKAT into KAT. Next, we generate from the latter a Moore automaton $\mathscr{X}_{\iota(e)}$ accepting $[\![e]\!]$, by using Kozen's syntactic Brzozowksi derivatives for KAT [91]. Finally, we answer an equivalence query from a Moore automaton $\mathscr{Y}$ by running a bisimulation between $\mathscr{X}_{\iota(e)}$ and $\mathscr{Y}$, similarly to [127, Fig. 1], and a membership query from $w\alpha \in \mathsf{GS}$ by returning the value of α at the output of the state in $\mathscr{X}_{\iota(e)}$ reached by w, that is, $[\![e]\!](w\alpha)$. A membership query from $w \in \mathsf{GS}^-$ is answered by querying $w\alpha \in \mathsf{GS}$ for all $\alpha \in \mathsf{At}$.

With the oracle for L^*, we can derive an oracle for GL^* as follows. Membership queries $w\alpha \in \mathsf{GS}$ are delegated and answered by the oracle for L^* as explained above. An equivalence query from a GKAT automaton $\mathscr{Y}$ is answered by posing an equivalence query to the oracle for L^* with the Moore automaton $f(\mathscr{Y})$ obtained via the embedding defined in Lemma 3.6.1.1. If the oracle for L^* replies with a counterexam-

	t_1	$\overline{t_1}$
ε	0	0
$\overline{t_1}p_2$	1	1
t_1p_1	1	1
t_1p_2	0	0
t_1p_3	0	0
$\overline{t_1}p_1$	0	0
$\overline{t_1}p_3$	0	0
$\overline{t_1}p_2t_1p_1$	0	0
$\overline{t_1}p_2t_1p_2$	0	0
$\overline{t_1}p_2t_1p_3$	0	0
$\overline{t_1}p_2\overline{t_1}p_1$	0	0
$\overline{t_1}p_2\overline{t_1}p_2$	0	0
$\overline{t_1}p_2\overline{t_1}p_3$	0	0

(a)



(b)

Figure 3.8: For $e = \texttt{if } t_1 \texttt{ then do } p_1 \texttt{ else do } p_2$, $|\Sigma| = 3$, and $|T| = 2$, our implementation of GL^* accepts the table in (a), which induces the automaton in (b).

ple $z \in \mathsf{GS}^-$, we extend z with an $\alpha \in \mathsf{At}$ such that $[\![\mathscr{Y}]\!](z\alpha) \neq [\![e]\!](z\alpha)$.

3.8 Related Work

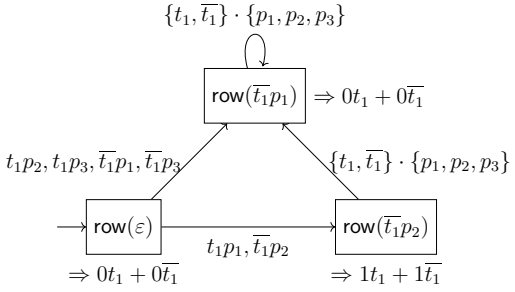
GKAT is a variation on KAT [92] that one obtains by restricting the union and iteration operations from KAT to guarded versions. While GKAT is less expressive than KAT, term equivalence is notably more efficiently decidable [142, 92], making it a candidate for the foundations of network-programming [143, 12, 57]

GKAT automata appear in the literature already prior to [142], e.g. in the work of Kozen [93] under the name *strictly deterministic automata*. In the latter, Kozen states that GKAT automata correspond to a limited class of *automata with guarded strings (AGS)* [89], for which he gives determinisation and minimisation constructions. In a different paper [91] Kozen introduces a second definition of (deterministic) AGS as Moore automata, and states the difference to the definition in [89] is inessential.

Recently, a new perspective on the semantics and coalgebraic theory of GKAT has been given in terms of coequations [135, 46]. Using the Thompson construction, it is possible to construct for every expression e a language equivalent automaton $\mathscr{X}_e$. In [93] it was shown that in general there is no reverse construction: there exists a

	ε	$\overline{t_1}p_1\overline{t_1}p_2$	$\overline{t_1}p_2$
ε	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$1t_1 + 1\overline{t_1}$
$\overline{t_1}p_1$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_2$	$1t_1 + 1\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
t_1p_1	$1t_1 + 1\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
t_1p_2	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
t_1p_3	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_1t_1p_1$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_1t_1p_2$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_1t_1p_3$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_1\overline{t_1}p_1$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_1\overline{t_1}p_2$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_1\overline{t_1}p_3$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_2t_1p_1$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_2t_1p_2$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_2t_1p_3$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_2\overline{t_1}p_1$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_2\overline{t_1}p_2$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$
$\overline{t_1}p_2\overline{t_1}p_3$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$	$0t_1 + 0\overline{t_1}$

(a)



(b)

Figure 3.9: For $e = \texttt{if}\ t_1\ \texttt{then do}\ p_1\ \texttt{else do}\ p_2$, $|\Sigma| = 3$, and $|T| = 2$, our implementation of L^* accepts the table in (a), which induces the automaton in (b).

GKAT automaton that is inequivalent to $\mathscr{X}_e$ for all expressions e. In consequence, [142] proposed a subclass of *well-nested* automata and showed that every finite well-nested automaton is bisimilar to $\mathscr{X}_e$ for some e. In [135] it was shown that well-nestedness is in fact too restrictive: there exists an automaton that is bisimilar to $\mathscr{X}_e$ for some e, but not well-nested. To capture the *full* class of automata exhibiting the behaviour of expressions, one has to extend the class of well-nested automata to the class of automata satisfying the *nesting coequation*, which forms a *covariety* [46] (cf. Proposition 3.4.1.8) .

Regular inference, or active automata learning, is a technique used for deriving a model from a black-box by interacting with it via observations. The original algorithm L^* by Angluin [14] learns deterministic finite automata, but since then has been extended to other classes of automata [16, 1, 112], including Moore automata, as we have seen in Section 1.3. Typically, algorithms such as L^* are designed to output for a given language a unique minimal acceptor. Not all classes admit a canonical minimal acceptor, for instance, learning non-deterministic models is a challenge [49, 31, 161, 66], as is shown in detail in Chapter 4.

3.9 Discussion and Future Work

We have presented GL^*, an algorithm for learning the GKAT automaton representation of a black-box, by observing its behaviour via queries to an oracle. We have shown that for every normal GKAT automaton there exists a unique size-minimal normal automaton, accepting the same language: its minimisation. We have identified the minimisation with an alternative but equivalent construction, and derived its preservation of the nesting coequation. A central result showed that if the oracle in GL^* is instantiated with the language accepted by a finite normal automaton, then GL^* terminates with its minimisation. A complexity analysis showed the advantage of GL^* over L^* for learning automata representations of GKAT programs in terms of membership queries. We discussed additional optimizations, and implemented GL^* and L^* in OCaml to compare their performances on example programs.

There are numerous directions in which the present work could be further explored. In Section 3.6.3 we introduced an optimization for GL^* which is inspired by Rivest and Schapire's counterexample handling method for L^* [130]. The *oberservation pack* algorithm for L^* [71] has successfully combined Rivest and Schapire's method with an efficient *discrimination tree* data structure [83]. The state-of-the-art *TTT*-algorithm [74] for L^* extends the former with discriminator finalization techniques. It thus is natural to ask whether for GL^* there exist similar data structures, potentially exploiting the deterministic nature of the languages accepted by GKAT automata.

While L^* has seen major improvements over the years and has inspired numerous variations for different types of transition systems, all approaches remain in common their focus on the *equivalence* of observations. The recently presented $\mathsf{L}^\sharp$ algorithm [154] takes a different perspective: it instead focuses on *apartness*, a constructive form of inequality. $\mathsf{L}^\sharp$ does not require data-structures such as observation tables or discrimination trees, instead operating directly on tree-shaped automata. It remains open whether a similar shift in perspective is feasible for GL^*.

There exist various domain-specific extensions of KAT (e.g. KAT+B! [60], NetKAT [12], ProbNetKAT [58]), and similar directions have been proposed for GKAT. It has been noted that GKAT is better fit for probabilistic domains than KAT, as it avoids mixing non-determinism with probabilities [143]. Due to its foundations in KAT,

NetKAT's decision procedure is PSPACE-complete, thus hindering verification to scale. In contrast to KAT, the equational theory of GKAT is decidable in (almost) linear time, making GKAT an interesting alternative candidate for the foundations of a SDN programming language like NetKAT. While there currently exists no explicit automata learning algorithm for NetKAT in the style of L^*, there is work closely related [141, 57]. Generally, we expect that in the future, for any extension of GKAT, there will be interest in developing the corresponding automata (learning) theories.

Chapter 4

Canonical Automata

In Chapter 3 we explored the automata theory of GKAT and presented GL^*, an algorithm for learning a GKAT automaton by observing the behaviour of a black-box. The design of GL^* started with us assigning to a GKAT automaton a language equivalent second automaton – its minimisation. The algorithm itself has been a derivation of this construction: for a given language, it iteratively refines approximations of the *canonical* minimal target model we have proven to exist. In this chapter, we will investigate canonical target models of more general type.

The classical powerset construction is a standard method that is used to convert a non-deterministic automaton into a deterministic one recognising the same language. In [140], the powerset construction has been lifted to a more general framework that converts an automaton with side-effects, given by a monad, into a deterministic automaton accepting the same language. The resulting automaton has additional algebraic properties, in the state space and transition structure, inherited from the monad. We will study the reverse construction and present a framework in which a deterministic automaton with additional algebraic structure over a given monad can be converted into an equivalent succinct automaton with side-effects. Apart from recovering examples from the literature, such as the canonical residual finite-state automaton and the átomaton, we discover a new canonical automaton for a regular language by relating the free vector space monad over the two element field to the neighbourhood monad. Finally, we show that every regular language satisfying a suitable property parametric in two monads admits a size-minimal succinct acceptor.

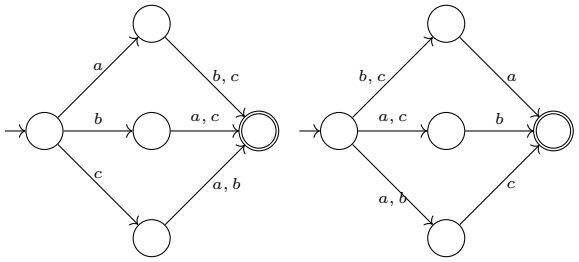


Figure 4.1: Two non-isomorphic size-minimal NFA accepting the language $\{ab, ac, ba, bc, ca, cb\} \subseteq \{a, b, c\}^*$ [20]

4.1 Introduction

The existence of a unique minimal *deterministic* acceptor is an important property of regular languages. Establishing a similar result for *non-deterministic* acceptors is of great practical importance, as non-deterministic automata can be exponentially more succinct than deterministic ones. Unfortunately, a regular language can be accepted by several size-minimal NFAs that are not isomorphic. In Figure 4.1 we give one example of such behaviour. A number of sub-classes of non-deterministic automata have been identified in the literature to tackle this issue, which all admit canonical representatives: the *átomaton* [39], the *canonical residual finite-state automaton* (short *canonical RFSA* and also known as *jiromaton*) [49], the *minimal xor automaton* [156], and the *distromaton* [119].

In this chapter we provide a general categorical framework that unifies constructions of canonical non-deterministic automata and unveils new ones. Our framework adopts the well-known representation of side-effects via monads (Definition 2.2.0.7) to generalise non-determinism in automata. For instance, an NFA (without initial states) can be represented as a pair (X, k), where X is the set of states and $k\colon X \to 2\times\mathcal{P}(X)^A$ combines the function classifying each state as accepting or rejecting with the function giving the set of next states for each input. The powerset forms a monad $(\mathcal{P}, \{-\}, \mu)$, where $\{-\}$ creates singleton sets and μ takes the union of a set of sets. This allows describing the classical powerset construction, converting an NFA into a DFA, in categorical terms as depicted on the left of Figure 4.2, where $k^\sharp\colon \mathcal{P}(X) \to 2 \times \mathcal{P}(X)^A$

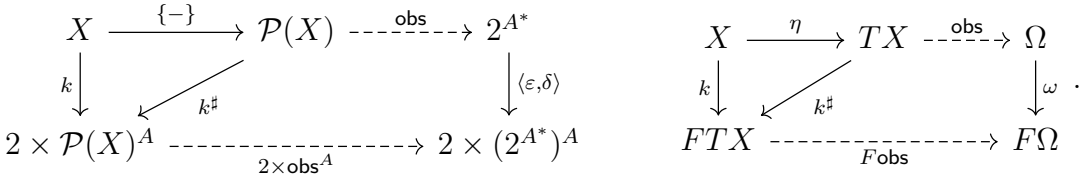


Figure 4.2: Generalised determinisation of automata with side-effects in a monad

represents an equivalent DFA[1], obtained by taking the subsets of X as states and $\langle \varepsilon, \delta \rangle : 2^{A^*} \to 2 \times (2^{A^*})^A$ is the automaton of languages. There then exists a unique automaton homomorphism obs, assigning a language semantics to each set of states.

As seen on the right of Figure 4.2 this perspective further enables a *generalised determinisation* construction [140], where $2 \times (-)^A$ is replaced by any (suitable) functor F describing the automaton structure, and $\mathcal{P}$ by a monad T describing the automaton side-effects. In this picture, $\Omega \xrightarrow{\omega} F\Omega$ is the final coalgebra (Definition 2.2.0.13), providing a semantic universe that generalises the automaton of languages.

Our work starts from the observation that the deterministic automata resulting from this generalised determinisation constructions have *additional algebraic structure*: the state space $\mathcal{P}(X)$ of the determinised automaton defines a free complete join-semilattice (CSL) over X, and $k^\sharp$ and obs are CSL homomorphisms. More generally, TX defines a (free) algebra for the monad T, and $k^\sharp$ and obs are T-algebra homomorphisms (Definition 2.2.0.9).

With this observation in mind, our question is: can we exploit the additional algebraic structure to "reverse" these constructions? In other words, can we convert a deterministic automaton with additional algebraic structure over a given monad to an equivalent succinct automaton with side-effects, possibly over another monad? To answer this question, we make the following contributions:

- We present a categorical framework based on bialgebras (Definition 4.3.0.8) and distributive law homomorphisms (Definition 4.5.1.1) that allows deriving canonical representatives for a wide class of succinct automata with side-effects in a monad.

[1]The classical powerset determinisation of $k = \langle \varepsilon, \delta \rangle : X \to 2 \times \mathcal{P}(X)^A$ is $k^\sharp = \langle \varepsilon^\sharp, \delta^\sharp \rangle : \mathcal{P}(X) \to 2 \times \mathcal{P}(X)^A$, where $\varepsilon^\sharp(U) = \vee_{x \in U} \varepsilon(x)$ and $\delta^\sharp(U)(a) = \cup_{x \in U} \delta(x)(a)$.

- We strictly improve the expressivity of previous work [67, 19]: our framework instantiates not only to well-known examples such as the canonical RFSA (Example 4.4.0.9) and the minimal xor automaton (Example 4.4.0.13), but also includes the átomaton (Section 4.5.3) and the distromaton (Section 4.5.4), which were not covered in [67, 19]. While other frameworks restrict themselves to the category of sets [67], we are able to include canonical acceptors in other categories, such as the *canonical nominal RFSA* (Example 4.4.0.12).

- We relate vector spaces over the unique two element field with complete atomic Boolean algebras and consequently discover a previously unknown canonical mod-2 weighted acceptor for regular languages—the *minimal xor-CABA automaton* (Section 4.5.5)—that in some sense is to the minimal xor automaton what the átomaton is to the canonical RFSA (Figure 4.10).

- We introduce an abstract notion of *closedness* for succinct automata that is parametric in two monads (Definition 4.6.0.2), and prove that every regular language satisfying a suitable property admits a canonical size-minimal representative among closed acceptors (Theorem 4.6.0.5). By instantiating the latter we subsume known minimality results for canonical automata, prove the xor-CABA automaton minimal, and establish a size comparison between different acceptors (Section 4.6.1).

4.2 Overview of the Approach

In this section, we give an overview of the ideas of this chapter through an example. We show how our methodology allows recovering the construction of the átomaton for the regular language $L = (a + b)^*a$, which consists of all words over $A = \{a, b\}$ that end in a. For each step, we hint at how it is generalised in our framework.

The classical construction of the átomaton for L consists in closing the *residuals*[2] of L under all Boolean operations, and then forming a non-deterministic au-

[2] A language is a *residual* or *left quotient* of $L \subseteq A^*$, if it is of the form $v^{-1}L = \{u \in A^* \mid vu \in L\}$ for some $v \in A^*$.

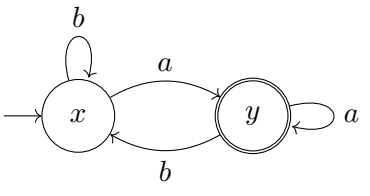


Figure 4.3: The minimal DFA for $L = (a + b)^* a$

tomaton whose states are the atoms[3] of the ensuing complete atomic[4] Boolean algebra (CABA)—that is, non-empty intersections of complemented or uncomplemented residuals. In our categorical setting, this construction is obtained in several steps, which we now describe.

4.2.1 Computing Residuals

We first construct the minimal DFA accepting L as a coalgebra of type $\mathsf{M}_L \to 2 \times (\mathsf{M}_L)^A$. By the well-known Myhill-Nerode theorem [120], M_L is the set of residuals for L. The automaton is depicted in Figure 4.3.

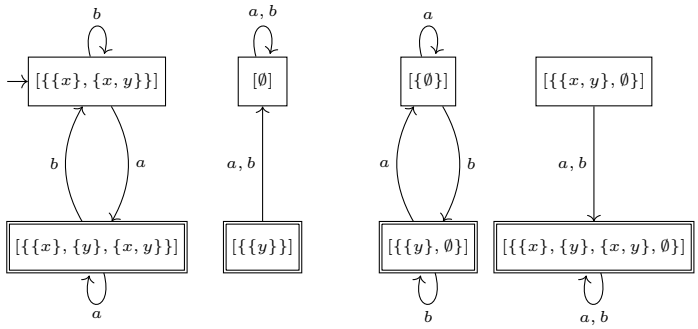
In our framework, we consider coalgebras over an arbitrary endofunctor $F \colon \mathscr{C} \to \mathscr{C}$ ($F = 2 \times (-)^A$ and $\mathscr{C} = \mathsf{Set}$ in this case), as introduced in Definition 2.2.0.12. Minimal realisations, generalising minimal DFAs, exist for a wide class of functors F and categories $\mathscr{C}$, including all the examples in this chapter.

4.2.2 Taking the Boolean Closure

We close the minimal DFA under all Boolean operations, generating an equivalent deterministic automaton that has additional algebraic structure: its state space is a CABA. This is achieved via a double powerset construction—where sets of sets are interpreted as full disjunctive normal form—and the resulting coalgebra is of type $\mathcal{P}^2(\mathsf{M}_L) \to 2 \times (\mathcal{P}^2(\mathsf{M}_L))^A$. Our construction relies on the *neighbourhood monad* $\mathcal{H}$ (Examples 2.2.0.8), whose algebras are precisely CABAs, and yields a (free) *bialgebra*

[3] A non-zero element a in a Boolean algebra B is called an *atom*, if for all $x \in B$ with $x \leq a$ it follows $x = 0$ or $x = a$.

[4] A Boolean algebra B is *atomic*, if for all $x \in B$ there exists a decomposition $x = \vee_I a_i$, where $\{a_i \mid i \in I\}$ is some set of atoms.



∧	1	2	3	4	5	6	7	8
1	1	2	2	1	1	2	2	1
2	2	2	2	2	2	2	2	2
3	2	2	3	3	2	2	3	3
4	1	2	3	4	1	2	3	4
5	1	2	2	1	5	6	6	5
6	2	2	2	2	6	6	6	6
7	2	2	3	3	6	6	7	7
8	1	2	3	4	5	6	7	8

	¬
1	7
2	8
3	5
4	6
5	3
6	4
7	1
8	2

Figure 4.4: The minimal CABA-structured DFA for $L = (a + b)^*a$, where $1 \equiv [\{\{x\}, \{x, y\}\}]$, $2 \equiv [\emptyset]$, $3 \equiv [\{\emptyset\}]$, $4 \equiv [\{\{x, y\}, \emptyset\}]$, $5 \equiv [\{\{x\}, \{y\}, \{x, y\}\}]$, $6 \equiv [\{\{y\}\}]$, $7 \equiv [\{\{y\}, \emptyset\}]$, $8 \equiv [\{\{x\}, \{y\}, \{x, y\}, \emptyset\}]$

capturing both the coalgebraic and the algebraic structure; the interplay of these two structures is captured via a *distributive law*. We then minimise this DFA to identify Boolean expressions evaluating to the same language. As desired, the resulting state space is precisely the Boolean closure of the residuals of L. Formally, we obtain the minimal bialgebra for L depicted in Figure 4.4.

This step in our framework is generalised as closure of an F-coalgebra w.r.t (the algebraic structured induced by) any monad S for which a suitable distributive law λ with the coalgebra endofunctor F exists. The first step of the closure yields a free λ-bialgebra, comprised of both an F-coalgebra and an S-algebra over the same state space. In a second step, minimisation is carried out in the category of λ-bialgebras, which guarantees simultaneous preservation of the algebraic structure and of the language semantics.

4.2.3 Constructing the Átomaton

This step is the key technical result of this chapter. Atoms have the property that their Boolean closure generates the entire CABA. In our framework, this property is generalised via the notion of *generators* for algebras over a monad, which allows one to represent a bialgebra as an equivalent *free* bialgebra over its generators, and hence to obtain succinct canonical representations (Proposition 4.4.0.5). In Section 4.4 we apply this result to obtain the canonical RFSA, the canonical nominal RFSA, and

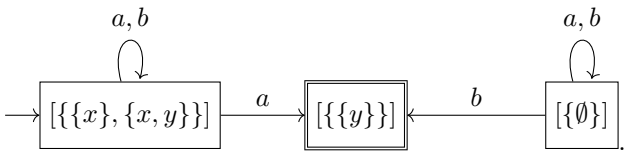


Figure 4.5: The átomaton for $L = (a + b)^* a$

the minimal xor automaton for a given regular language.

However, to recover the átomaton from the minimal CABA-structured DFA of the previous step, in addition a subtle change of perspective is required. In fact, we are still working with the "wrong" side-effect: the non-determinism of bialgebras so far is determined by $\mathcal{H}$, whereas we are interested in an NFA, whose non-determinism is captured by the powerset monad $\mathcal{P}$. As is well-known, every element of a CABA can be obtained as the join of the atoms below it. In other words, those atoms are also generators of the underlying CSL, which is an algebra for $\mathcal{P}$. We formally capture this idea as a map between monads $\mathcal{H} \to \mathcal{P}$. Crucially, we show that this map lifts to a *distributive law homomorphism* and allows translating a bialgebra over $\mathcal{H}$ to a bialgebra over $\mathcal{P}$, which can be represented as a free bialgebra over atoms—the átomaton for L, which is shown in Figure 4.5.

In Section 4.5 we generalise this idea to the situation of two monads S and T involved in distributive laws with the coalgebra endofunctor F. In particular, Corollary 4.5.1.3 is our free representation result, spelling out a condition under which a bialgebra over S can be represented as a free bialgebra over T, and hence admits an equivalent succinct representation as an automaton with side-effects in T. Besides the átomaton and the examples in Section 4.4, this construction allows us to capture the distromaton and a newly discovered canonical acceptor that relates CABAs with vector spaces over the two element field.

4.3 Distributive Laws and Bialgebras

In this section, we briefly recall distributive laws and bialgebras, which form the technical foundations of our category-theoretical framework.

Distributive laws have originally occurred as a way to compose monads [25], but

now also exist in a wide range of other forms [147]. For our particular case it is sufficient to consider distributive laws between a monad and an endofunctor, sometimes referred to as *Eilenberg-Moore laws* [81].

Definition 4.3.0.1 (Distributive Law)**.** A *distributive law* between a monad T and an endofunctor F on $\mathscr{C}$ is a natural transformation $\lambda : TF \Rightarrow FT$ such that the following two diagrams commute:

$$\begin{array}{ccc} FX & \xrightarrow{F\eta_X} & FTX \\ {\scriptstyle TFX}\downarrow & \nearrow_{\lambda_X} & \\ TFX & & \end{array} \qquad \begin{array}{ccccc} T^2FX & \xrightarrow{T\lambda_X} & TFTX & \xrightarrow{\lambda_{TX}} & FT^2X \\ {\scriptstyle \mu_{FX}}\downarrow & & & & \downarrow{\scriptstyle F\mu_X} \\ TFX & & \xrightarrow[\lambda_X]{} & & FTX \end{array}.$$

We are particularly interested in canonically induced distributive laws involving the endofunctor F with $FX = B \times X^A$. The following statement is well-known and appears for instance in [75]. We include a proof of the result, since we were unable to locate one in the literature. Note that the result can easily be generalised to a strong monad on any cartesian closed category.

Lemma 4.3.0.2. *Every algebra $h : TB \to B$ for a set monad T induces a distributive law λ^h between T and F with $FX = B \times X^A$ defined as the composition*

$$\lambda^h_X := T(B \times X^A) \overset{\langle T\pi_1, T\pi_2\rangle}{\longrightarrow} T(B) \times T(X^A) \overset{h\times \mathsf{st}}{\longrightarrow} B \times (TX)^A, \tag{4.1}$$

where st *denotes the canonical strength function*[5]*.*

Proof. The naturality of λ^h essentially follows from the naturality of the strength function. The equation involving the monad unit is a consequence of

$$\begin{aligned} & \pi_1 \circ (h \times \mathsf{st}) \circ \langle T\pi_1, T\pi_2\rangle \circ \eta_{B\times X^A} & \\ = \; & h \circ T\pi_1 \circ \eta_{B\times X^A} & \text{(Definition of } \pi_1\text{)} \\ = \; & h \circ \eta_B \circ \pi_1 & \text{(Naturality of } \eta\text{)} \\ = \; & \pi_1 \circ (B \times \eta^A_X) & \text{(Definition of } \pi_1, h \circ \eta_B = \mathsf{id}_B\text{)} \end{aligned}$$

[5]For any two sets X, A the strength function $\mathsf{st} : T(X^A) \to (TX)^A$ is defined by $\mathsf{st}(U)(a) = T(\mathsf{ev}_a)(U)$, where $\mathsf{ev}_a(f) = f(a)$.

and the chain of equalities

$$\begin{aligned}
\pi_2 \circ (h \times \mathsf{st}) \circ \langle T\pi_1, T\pi_2 \rangle \circ \eta_{B \times X^A} &= \mathsf{st} \circ T\pi_2 \circ \eta_{B \times X^A} && \text{(Definition of } \pi_2\text{)} \\
&= \mathsf{st} \circ \eta_{X^A} \circ \pi_2 && \text{(Naturality of } \eta\text{)} \\
&= \eta_X^A \circ \pi_2 && (\mathsf{st} \circ \eta_{X^A} = \eta_X^A) \\
&= \pi_2 \circ (B \times \eta_X^A) && \text{(Definition of } \pi_2\text{)}
\end{aligned}$$

Similarly, the equation involving the monad multiplication is a consequence of

$$\begin{aligned}
&\pi_1 \circ (B \times \mu_X^A) \circ (h \times \mathsf{st}) \circ \langle T\pi_1, T\pi_2 \rangle \circ T(h \times \mathsf{st}) \circ T\langle T\pi_1, T\pi_2 \rangle \\
&= \text{(Definition of } \pi_1\text{)} \\
&\quad h \circ Th \circ T^2\pi_1 \\
&= (h \circ Th = h \circ \mu_B) \\
&\quad h \circ \mu_B \circ T^2\pi_1 \\
&= \text{(Naturality of } \mu\text{)} \\
&\quad h \circ T\pi_1 \circ \mu_{B \times X^A} \\
&= \text{(Definition of } \pi_1\text{)} \\
&\quad \pi_1 \circ (h \times \mathsf{st}) \circ \langle T\pi_1, T\pi_2 \rangle \circ \mu_{B \times X^A}
\end{aligned}$$

and the equalities

$$\begin{aligned}
&\pi_2 \circ (B \times \mu_X^A) \circ (h \times \mathsf{st}) \circ \langle T\pi_1, T\pi_2 \rangle \circ T(h \times \mathsf{st}) \circ T(T\pi_1, T\pi_2) \\
&= \text{(Definition of } \pi_2\text{)} \\
&\quad \mu_X^A \circ \mathsf{st} \circ T(\mathsf{st}) \circ T^2\pi_2 \\
&= (\mu_X^A \circ \mathsf{st} \circ T(\mathsf{st}) = \mathsf{st} \circ \mu_{X^A}) \\
&\quad \mathsf{st} \circ \mu_{X^A} \circ T^2\pi_2 \\
&= \text{(Naturality of } \mu\text{)} \\
&\quad \mathsf{st} \circ T\pi_2 \circ \mu_{B \times X^A} \\
&= \text{(Definition of } \pi_2\text{)} \\
&\quad \pi_2 \circ (h \times \mathsf{st}) \circ \langle T\pi_1, T\pi_2 \rangle \circ \mu_{B \times X^A}
\end{aligned}$$

□

Of particular importance for us are the canonical algebra structures for the output set $B = 2$. For instance, the algebra structures defined by $h^{\mathcal{P}}(\varphi) = h^{\mathcal{R}}(\varphi) = \varphi(1)$ and $h^{\mathcal{H}}(\Phi) = h^{\mathcal{A}}(\Phi) = \Phi(\mathsf{id}_2)$, where we identify subsets with their characteristic functions, and $\mathcal{R}$ and $\mathcal{A}$ denote the free vector space monad over the unique two element field $\mathbb{Z}_2$, and the monotone neighbourhood monad, respectively (cf. Examples 2.2.0.8). In these cases we will abuse notation and write λ^T instead of λ^{h^T}. The soundness of the definitions is witnessed by the following four short results.

Lemma 4.3.0.3. *The morphism $h^{\mathcal{P}} : \mathcal{P}2 \to 2$ satisfying $\varphi \mapsto \varphi(1)$ defines a $\mathcal{P}$-algebra.*

Proof. On the one hand we find

$$
\begin{aligned}
& h^{\mathcal{P}} \circ \mu_2^{\mathcal{P}}(\Phi) \\
&= \mu_2^{\mathcal{P}}(\Phi)(1) && \text{(Definition of } h^{\mathcal{P}}) \\
&= \textstyle\bigvee_{\varphi \in 2^2} \Phi(\varphi) \wedge \varphi(1) && \text{(Definition of } \mu_2^{\mathcal{P}}) \\
&= (\textstyle\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(1)} \Phi(\varphi) \wedge \varphi(1)) \vee (\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(0)} \Phi(\varphi) \wedge \varphi(1)) && \text{(Case split)} \\
&= (\textstyle\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(1)} \Phi(\varphi) \wedge \varphi(1)) \vee (\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(0)} \Phi(\varphi) \wedge 0) && (\varphi \in (h^{\mathcal{P}})^{-1}(0)) \\
&= (\textstyle\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(1)} \Phi(\varphi) \wedge \varphi(1)) \vee (\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(0)} 0) && (a \wedge 0 = 0) \\
&= (\textstyle\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(1)} \Phi(\varphi) \wedge \varphi(1)) \vee 0 && (0 \vee 0 = 0) \\
&= \textstyle\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(1)} \Phi(\varphi) \wedge \varphi(1) && (a \vee 0 = a) \\
&= \textstyle\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(1)} \Phi(\varphi) \wedge 1 && (\varphi \in (h^{\mathcal{P}})^{-1}(1)) \\
&= \textstyle\bigvee_{\varphi \in (h^{\mathcal{P}})^{-1}(1)} \Phi(\varphi) && (a \wedge 1 = a) \\
&= \mathcal{P}(h^{\mathcal{P}})(\Phi)(1) && \text{(Definition of } \mathcal{P}(h^{\mathcal{P}})) \\
&= h^{\mathcal{P}} \circ \mathcal{P}(h^{\mathcal{P}})(\Phi) && \text{(Definition of } h^{\mathcal{P}})
\end{aligned}
$$

and on the other hand we can deduce

$$\begin{aligned} h^{\mathcal{P}} \circ \eta^{\mathcal{P}}_2(x) &= \eta^{\mathcal{P}}_2(x)(1) && \text{(Definition of } h^{\mathcal{P}}\text{)} \\ &= [x = 1] && \text{(Definition of } \eta^{\mathcal{P}}_2\text{)} \\ &= x && \text{(Definition of } [\cdot]\text{)} \end{aligned}$$

where $[x = y]$ is 1, if $x = y$, and 0 otherwise. □

As one verifies, under the equivalence $\mathsf{Set}^{\mathcal{P}} \cong \mathsf{CSL}$, the morphism $h^{\mathcal{P}}$ equips 2 with its canonical complete join-semilattice structure $(2, \vee)$.

Lemma 4.3.0.4. *The morphism $h^{\mathcal{H}} : \mathcal{H}2 \to 2$ assigning $\Phi \mapsto \Phi(\mathsf{id}_2)$ defines an $\mathcal{H}$-algebra.*

Proof. Since $\eta^{\mathcal{H}}_{2^2}(\mathsf{id}_2)(\Phi) = \Phi(\mathsf{id}_2) = h^{\mathcal{H}}(\Phi)$ we find

$$\begin{aligned} h^{\mathcal{H}} \circ \mu^{\mathcal{H}}_2(\Psi) &= \mu^{\mathcal{H}}_2(\Psi)(\mathsf{id}_2) && \text{(Definition of } h^{\mathcal{H}}\text{)} \\ &= \Psi(\eta^{\mathcal{H}}_{2^2}(\mathsf{id}_2)) && \text{(Definition of } \mu^{\mathcal{H}}_2\text{)} \\ &= \Psi(\mathsf{id}_2 \circ h^{\mathcal{H}}) && (\eta^{\mathcal{H}}_{2^2}(\mathsf{id}_2) = h^{\mathcal{H}}) \\ &= \mathcal{H}(h^{\mathcal{H}})(\Psi)(\mathsf{id}_2) && \text{(Definition of } \mathcal{H}(h^{\mathcal{H}})\text{)} \\ &= h^{\mathcal{H}} \circ \mathcal{H}(h^{\mathcal{H}})(\Psi) && \text{(Definition of } h^{\mathcal{H}}\text{)} \end{aligned}$$

We further can deduce

$$\begin{aligned} h^{\mathcal{H}} \circ \eta^{\mathcal{H}}_X(x) &= \eta^{\mathcal{H}}_X(x)(\mathsf{id}_2) && \text{(Definition of } h^{\mathcal{H}}\text{)} \\ &= \mathsf{id}_2(x) && \text{(Definition of } \eta^{\mathcal{H}}_X\text{)} \\ &= x && \text{(Definition of } \mathsf{id}_2\text{)} \end{aligned}$$

□

As one verifies, under the equivalence $\mathsf{Set}^{\mathcal{H}} \cong \mathsf{CABA}$, the morphism $h^{\mathcal{H}}$ equips 2 with its canonical complete atomic Boolean algebra structure $(2, \vee, \wedge, \neg)$.

Lemma 4.3.0.5. *The morphism $h^{\mathcal{A}} : \mathcal{A}2 \to 2$ assigning $\Phi \mapsto \Phi(\mathsf{id}_2)$ defines an $\mathcal{A}$-algebra.*

Proof. Analogous to the proof of Lemma 4.3.0.4. □

As one verifies, under the equivalence $\mathsf{Set}^{\mathcal{A}} \cong \mathsf{CDL}$, the morphism $h^{\mathcal{A}}$ equips 2 with its canonical completely distributive lattice structure $(2, \vee, \wedge)$.

Lemma 4.3.0.6. *The morphism $h^{\mathcal{R}} : \mathcal{R}2 \to 2$ satisfying $\varphi \mapsto \varphi(1)$ defines an $\mathcal{R}$-algebra.*

Proof. Analogous to the proof of Lemma 4.3.0.3, after observing that $a \wedge 0 = 0$, $0 \oplus 0 = 0$, $a \oplus 0 = a$ and $a \wedge 1 = a$ for all $a \in 2$. □

As one verifies, under the equivalence $\mathsf{Set}^{\mathcal{R}} \cong \mathbb{Z}_2\text{-}\mathsf{Vect}$, the morphism $h^{\mathcal{R}}$ equips 2 with its canonical $\mathbb{Z}_2$-vector space structure $\mathbb{Z}_2 \cong (2, \oplus, \wedge)$.

Example 4.3.0.7 (Generalised Determinisation [133])**.** Given a distributive law, one can model the determinisation of a system with dynamics in F and side-effects in T (sometimes referred to as *succinct* automaton) by lifting an FT-coalgebra (X, k) to the F-coalgebra $(TX, k^\sharp)$, where $k^\sharp := (F\mu_X \circ \lambda_{TX}) \circ Tk$. As one verifies, the latter is in fact a T-algebra homomorphism of type $k^\sharp : (TX, \mu_X) \to (FTX, F\mu_X \circ \lambda_{TX})$. For instance, if the distributive law λ is induced by the disjunctive $\mathcal{P}$-algebra $h^{\mathcal{P}} : \mathcal{P}2 \to 2$ with $h^{\mathcal{P}}(\varphi) = \bigvee_{u \in \varphi} u = \varphi(1)$, the lifting $k^\sharp$ is the DFA in CSL obtained from an NFA k via the classical powerset construction.

The example above illustrates the concept of a bialgebra: the algebraic part (TX, μ_X) and the coalgebraic part $(TX, k^\sharp)$ of a lifted automaton are compatible along the distributive law λ.

Definition 4.3.0.8 (Bialgebra)**.** A *λ-bialgebra* is a tuple (X, h, k) consisting of a T-algebra (X, h) and an F-coalgebra (X, k) such that the following diagram commutes:

$$\begin{array}{ccccc} TX & & \xrightarrow{\quad h \quad} & & X \\ {\scriptstyle Tk}\downarrow & & & & \downarrow{\scriptstyle k} \\ TFX & \xrightarrow[\lambda_X]{} & FTX & \xrightarrow[Fh]{} & FX \end{array}$$

A homomorphism between λ-bialgebras is a morphism between the underlying objects that is simultaneously a T-algebra homomorphism and an F-coalgebra homomorphism. The category of λ-bialgebras and homomorphisms is denoted by $\mathsf{Bialg}(\lambda)$.

The existence of a final F-coalgebra is equivalent to the existence of a final λ-bialgebra, as the next result shows.

Lemma 4.3.0.9 ([81])**.** *Let* (Ω, k_Ω) *be the final* F*-coalgebra, then* $(\Omega, h_\Omega, k_\Omega)$ *with* $h_\Omega := \mathsf{obs}_{(T\Omega, \lambda_\Omega \circ Tk_\Theta)}$ *is the final* λ*-bialgebra satisfying* $\mathsf{obs}_{(X,h,k)} = \mathsf{obs}_{(X,k)}$*. Conversely, if* $(\Omega, h_\Omega, k_\Omega)$ *is the final* λ*-bialgebra, then* (Ω, k_Ω) *is the final* F*-coalgebra.*

For the distributive law in Example 4.3.0.7, the final bialgebra is carried by the final coalgebra $\mathcal{P}(A^*)$ with the free $\mathcal{P}$-algebra structure that takes the union of languages.

The generalised determinisation in Example 4.3.0.7 can be rephrased as a functor exp_T that *expands* an F-coalgebra with side-effects in T into a λ-bialgebra.

Lemma 4.3.0.10 ([81])**.** *Defining* $\mathsf{exp}_T(X,k) := (TX, \mu_X, (F\mu_X \circ \lambda_{TX}) \circ Tk)$ *and* $\mathsf{exp}_T(f) := Tf$ *yields a functor* $\mathsf{exp}_T : \mathsf{Coalg}(FT) \to \mathsf{Bialg}(\lambda)$*.*

We will also refer to the functor free_T that arises from exp_T by pre-composition with the canonical embedding of F-coalgebras into FT-coalgebras, therefore assigning to an F-coalgebra the λ-bialgebra it *freely* generates.

Corollary 4.3.0.11. *Defining* $\mathsf{free}_T(X,k) := (TX, \mu_X, \lambda_X \circ Tk)$ *and* $\mathsf{free}_T(f) := Tf$ *yields a functor* $\mathsf{free}_T : \mathsf{Coalg}(F) \to \mathsf{Bialg}(\lambda)$ *with* $\mathsf{free}_T(X,k) = \mathsf{exp}_T(X, F\eta_X \circ k)$*.*

4.4 Succinct Automata from Bialgebras

In this section we discuss the foundations of our theoretical contributions. We begin by recalling the notion of a *generator* [19] and demonstrate how it can be used to translate a bialgebra into an equivalent free bialgebra. While the treatment is very general, we are particularly interested in the case in which the bialgebra is given by a deterministic automaton that has additional algebraic structure over a given monad, and the translation results in an automaton with side-effects in that monad. We will demonstrate that the theory in this section instantiates to the canonical RFSA [49], the canonical *nominal* RFSA [111], and the minimal xor automaton [156].

Definition 4.4.0.1 (Generator and Basis)**.** A *generator* for a T-algebra (X, h) is a tuple (Y, i, d) consisting of an object Y, a morphism $i\colon Y \to X$, and a morphism $d\colon X \to TY$ such that $(h \circ Ti) \circ d = \mathsf{id}_X$. A generator is called a *basis* if it additionally satisfies $d \circ (h \circ Ti) = \mathsf{id}_{TY}$, that is, the following two diagrams commute:

$$\begin{array}{ccc} X & \xrightarrow{\mathsf{id}_X} & X \\ {\scriptstyle d}\downarrow & & \uparrow{\scriptstyle h} \\ TY & \xrightarrow[Ti]{} & TX \end{array} \qquad \begin{array}{ccc} TY & \xrightarrow{\mathsf{id}_{TY}} & TY \\ {\scriptstyle Ti}\downarrow & & \uparrow{\scriptstyle d} \\ TX & \xrightarrow[h]{} & X \end{array} .$$

A generator for an algebra is called a *scoop* by Arbib and Manes [19]. Here, we additionally introduce an abstract perspective on the notion of a basis. Intuitively, one calls a set Y that is embedded into an algebraic structure X a generator for the latter if every element $x \in X$ admits a decomposition $d(x) \in TY$ into a formal combination of elements of Y that evaluates to x. If the decomposition is moreover *unique*, that is, $h \circ Ti$ is not only a *surjection* with right-inverse d, but a *bijection* with two-sided inverse d, then a generator is called a basis. Every algebra is generated by itself using the generator $(X, \mathsf{id}_X, \eta_X)$ and every free algebra (TY, μ_Y) admits the basis $(Y, \eta_Y, \mathsf{id}_{TY})$. In fact, an algebra admits a basis if and only if it is isomorphic to a free algebra (cf. Lemma 4.4.0.7). We are particularly interested in classes of set-based algebras for which *every* algebra admits a *size-minimal* generator, that is, no generator has a carrier of smaller size. In such a situation we will also speak of *canonical* generators.

Example 4.4.0.2.

- A tuple (Y, i, d) is a generator for a $\mathcal{P}$-algebra $L = (X, h) \simeq (X, \vee^h)$ iff $x = \vee^h_{y \in d(x)} i(y)$ for all $x \in X$, where we write $\vee^h$ for the complete join-semilattice structure induced by h. Note that if $Y \subseteq X$ is a subset, then $i(y) = y$ for all $y \in Y$. If L satisfies the descending chain condition[6] (DCC), which is in particular the case if X is finite, then defining $i(y) = y$ and $d(x) = \{y \in J(L) \mid y \leq x\}$ turns the set of join-irreducibles[7] $J(L)$ into a size-minimal generator $(J(L), i, d)$ for L, cf. Lemma 4.4.0.3 below.

[6] A partially ordered set $(P, \leq)$ satisfies the *descending chain condition* if any descending chain $a_0 \geq a_1 \geq a_2 \geq \ldots$ in P stabilises, that is, there exists some $n \geq 0$, such that $a_n = a_{n+k}$ for all $k \geq 0$.

[7] A non-zero element a in a lattice L is called *join-irreducible*, if for all $y, z \in L$ with $a = y \vee z$ it follows $a = y$ or $a = z$.

- A tuple (Y, i, d) is a generator for an $\mathcal{R}$-algebra $V = (X, h) \simeq (X, +^h, \cdot^h)$ iff $x = \sum_{y \in Y}^h d(x)(y) \cdot^h i(y)$ for all $x \in X$, where we write $+^h$ and $\cdot^h$ for the $\mathbb{Z}_2$-vector space structure induced by h. As it is well-known that every vector space can be equipped with a basis, every $\mathcal{R}$-algebra V admits a basis. One can show that any finite basis is a size-minimal generator, cf. Lemma 4.4.0.4 below.

Lemma 4.4.0.3. *For any finite $\mathcal{P}$-algebra $L = (X, h)$ the join-irreducibles $(J(L), i, d)$ with $i(y) = y$ and $d(x) = \{y \in J(L) \mid y \leq x\}$ constitute a size-minimal generator.*

Proof. Since L is finite, it satisfies the DCC, which in turn can be used to show that the join-irreducibles constitute a generator as follows.

Assume there exists some $x \in X$ with $x \neq i^\sharp(d(x))$. We build an infinite sequence (a_n) with $a_i > a_{i+1}$ and $a_i \neq i^\sharp(d(a_i))$, which contradicts the DCC. For the base case we define $a_0 := x$. For any $x \in X$, the property $x \in J(L)$ immediately implies $x = i^\sharp(d(x))$. Thus we can assume $a_i \notin J(L)$. In consequence we have $a_i = y \vee z$ for $y, z \neq a_i$, i.e $a_i > y$ and $a_i > z$. Assume $y = i^\sharp(d(y))$ and $z = i^\sharp(d(z))$. Then

$$i^\sharp(d(a_i)) \leq a_i = y \vee z = i^\sharp(d(y)) \vee i^\sharp(d(z)) = i^\sharp(d(y) \vee d(z)) \leq i^\sharp(d(a_i)).$$

It thus follows $a_i = i^\sharp(d(a_i))$, which is a contradiction. Hence, w.l.o.g. assume $y \neq i^\sharp(d(y))$ and define $a_{i+1} := y$.

Let (Y, i', d') be an arbitrary generator for L. For any $a \in J(L)$ we have $a = \bigvee_{y \in d'(a)}^h i'(y)$. By the definition of join-irreducibles there exists at least one $y_a \in d'(a)$ such that $i'(y_a) = a$. One can thus define a function $f : J(L) \to Y$ with $f(a) = y_a$. Assume $f(a) = f(b)$, i.e. $y_a = y_b$. Then, by definition,

$$a = i'(y_a) = i'(y_b) = b$$

which shows that f is injective. It thus follows $|J(L)| \leq |Y|$. □

Lemma 4.4.0.4. *Any finite basis for an $\mathcal{R}$-algebra is a size-minimal generator.*

Proof. Let (Y, i, d) be a basis for an $\mathcal{R}$-algebra (X, h) with Y being finite. It immediately follows that $\mathcal{R}Y = 2^Y$. Let (Y', i', d') be any other generator for (X, h). We can assume that Y' is finite, as otherwise Y would be immediately of smaller size. It thus

follows that $\mathcal{R}Y' = 2^{Y'}$. By definition, $d : X \to \mathcal{R}Y$ is a bijection with inverse $h \circ \mathcal{R}i$, and $h \circ \mathcal{R}i' : \mathcal{R}Y' \to X$ is a surjection. We thus know that $d \circ h \circ \mathcal{R}i' : \mathcal{R}Y' \to \mathcal{R}Y$ is a surjection. In consequence we find that $|\mathcal{R}Y| \leq |\mathcal{R}Y'|$. Assume $|Y'| < |Y|$, then we can deduce

$$|\mathcal{R}Y'| = |2^{Y'}| = |2|^{|Y'|} < |2|^{|Y|} = |2^{Y}| = |\mathcal{R}Y|,$$

which contradicts $|\mathcal{R}Y| \leq |\mathcal{R}Y'|$. We thus can conclude that $|Y'| \geq |Y|$. □

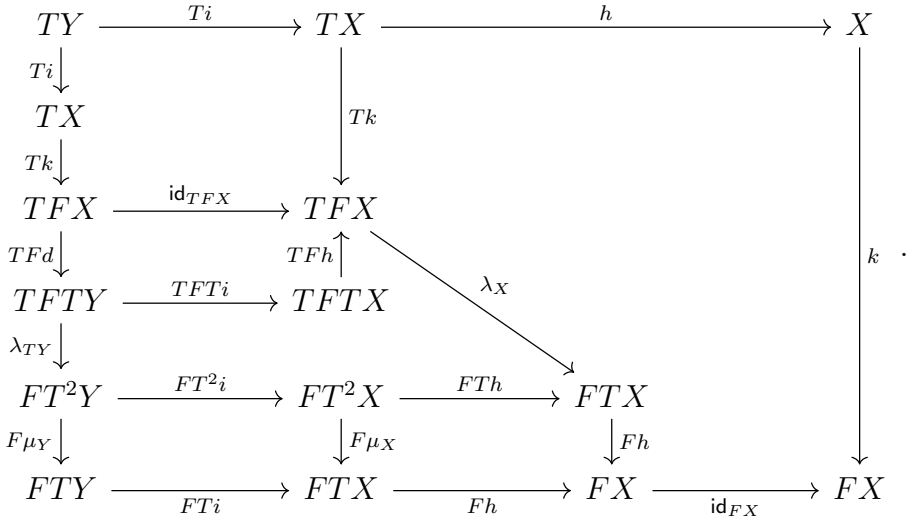
It is enough to find generators for the underlying algebra of a bialgebra to derive an equivalent free bialgebra. This is because the algebraic and coalgebraic components are tightly intertwined via a distributive law.

Proposition 4.4.0.5. *Let (X, h, k) be a λ-bialgebra and let (Y, i, d) be a generator for the T-algebra (X, h). Then $h \circ Ti \colon \mathsf{exp}_T((Y, Fd \circ k \circ i)) \to (X, h, k)$ is a λ-bialgebra homomorphism.*

Proof. By definition we have the equality

$$\mathsf{exp}_T((Y, Fd \circ k \circ i)) = (TY, \mu_Y, F\mu_Y \circ \lambda_{TY} \circ T(Fd \circ k \circ i)).$$

It is well-known that $h \circ Ti$ is a homomorphism between the underlying T-algebra structures. It thus remains to show that it is an F-coalgebra homomorphism. The latter follows from the commutativity of the diagram below:



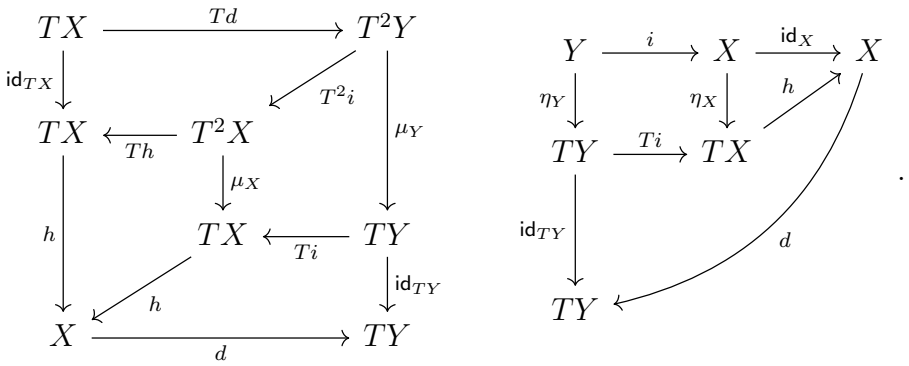
□

Intuitively, the bialgebra (X, h, k) is a deterministic automaton with additional algebraic structure in the monad T and say initial state $x \in X$, while the equivalent free bialgebra is the determinisation of the succinct automaton $Fd \circ k \circ i \colon Y \to FTY$ with side-effects in T and initial state $d(x) \in TY$.

The statement below establishes that, for bases, the morphism d is an algebra homomorphism, and, intuitively, that elements of a basis are uniquely generated by their image under the monad unit, that is, typically by themselves. We will use this technical result, among others, in Proposition 4.4.0.8.

Lemma 4.4.0.6. *Let (Y, i, d) be a basis for a T-algebra (X, h). Then $\mu_Y \circ Td = d \circ h$ and $d \circ i = \eta_Y$.*

Proof. The statement follows from the commutativity of the following two diagrams:



Alternatively, the first equality can be deduced from Beck's monadicity theorem: since the forgetful functor $U : \mathscr{C}^T \to \mathscr{C}$ is monadic, it reflects isomorphisms. □

Lemma 4.4.0.7. *A T-algebra admits a basis iff it is isomorphic to a free T-algebra.*

Proof. Assume (X, h) admits a basis (Y, i, d). Then, by definition, $d : X \to TY$ and $h \circ Ti : TY \to X$ are inverse to each other, thus witnessing an isomorphism of X and TY in the base category $\mathscr{C}$. It remains to show that the isomorphism lifts to $\mathscr{C}^T$. It immediately follows that $h \circ Ti$ lifts to a T-algebra homomorphism $h \circ Ti : (TY, \mu_Y) \to (X, h)$. The morphism d lifts to a T-algebra homomorphism $d : (X, h) \to (TY, \mu_Y)$ by Lemma 4.4.0.6.

Conversely, assume $f : (X, h) \to (TY, \mu_Y)$ is an isomorphism in $\mathscr{C}^T$. Let $i := f^{-1} \circ \eta_Y : Y \to X$ and $d := f : X \to TY$. Then (Y, i, d) is a basis for (X, h), since

$$\begin{aligned} d \circ (h \circ Ti) &= f \circ h \circ T(f^{-1} \circ \eta_Y) && \text{(Definitions of } i, d\text{)} \\ &= f \circ h \circ T(f^{-1}) \circ T(\eta_Y) && \text{(Functoriality of } T\text{)} \\ &= f \circ f^{-1} \circ \mu_Y \circ T(\eta_Y) && (f^{-1} \text{ is homomorphism}) \\ &= f \circ f^{-1} && (\mu_Y \circ T(\eta_Y) = \mathsf{id}_{TY}) \\ &= \mathsf{id}_{TY} && (f \text{ is isomorphism}) \end{aligned}$$

and similarly

$$\begin{aligned} (h \circ Ti) \circ d &= h \circ T(f^{-1} \circ \eta_Y) \circ f && \text{(Definitions of } i, d\text{)} \\ &= h \circ T(f^{-1}) \circ T(\eta_Y) \circ f && \text{(Functoriality of } T\text{)} \\ &= f^{-1} \circ \mu_Y \circ T(\eta_Y) \circ f && (f^{-1} \text{ is homomorphism}) \\ &= f^{-1} \circ f && (\mu_Y \circ T(\eta_Y) = \mathsf{id}_{TY}) \\ &= \mathsf{id}_X && (f \text{ is isomorphism}) \end{aligned}$$

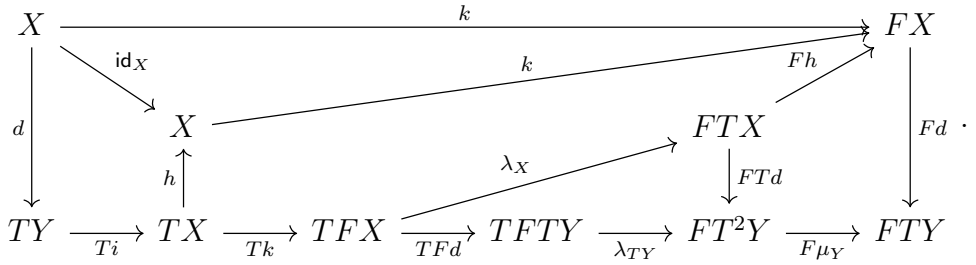
□

The following result observes that if the generator for the algebra of a bialgebra in Proposition 4.4.0.5 is in fact a *basis*, then the equivalence involved is an isomorphism.

Proposition 4.4.0.8. *Let (X, h, k) be a λ-bialgebra and let (Y, i, d) be a basis for the T-algebra (X, h). Then $h \circ Ti \colon \mathsf{exp}_T((Y, Fd \circ k \circ i)) \to (X, h, k)$ is a λ-bialgebra isomorphism.*

Proof. From Proposition 4.4.0.5 we know that $h \circ Ti$ is a λ-bialgebra homomorphism. By the definition of a basis, d is a two-sided inverse to $h \circ Ti$ as ordinary morphism. It thus remains to show that d is a λ-bialgebra homomorphism. By Lemma 4.4.0.6 it is a T-algebra homomorphism, and the diagram below shows that it commutes with

F-coalgebra structures:



□

We conclude this section by illustrating how Proposition 4.4.0.5 can be used to construct the canonical RFSA [49], the canonical nominal RFSA [111], and the minimal xor automaton [156] for a regular language L over some alphabet A. All examples follow three analogous steps:

1. We construct the minimal[8] pointed coalgebra M_L for the (nominal) set endofunctor $F = 2 \times (-)^A$ accepting L. For the case $A = \{a, b\}$ and $L = (a + b)^* a$, the coalgebra M_L is depicted in Figure 4.3.

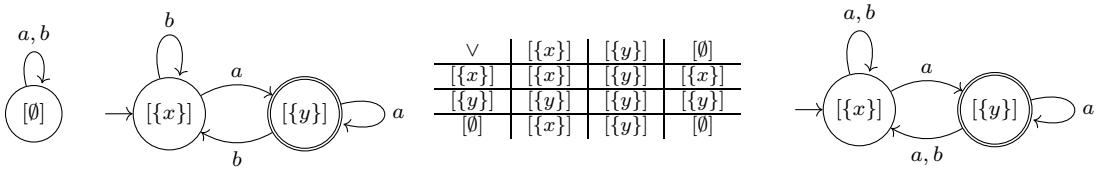
2. We equip the former with additional algebraic structure in a monad T (which is related to F via a canonically induced distributive law λ) by generating the λ-bialgebra $\mathsf{free}_T(\mathsf{M}_L)$. By identifying semantically equivalent states we consequently derive the minimal[9] (pointed) λ-bialgebra (X, h, k) for L.

3. We identify canonical generators (Y, i, d) for (X, h) and use Proposition 4.4.0.5 to derive an equivalent succinct automaton $(Y, Fd \circ k \circ i)$ with side-effects in T.

Example 4.4.0.9 (The Canonical RFSA)**.** Using the $\mathcal{P}$-algebra structure $h^{\mathcal{P}} : \mathcal{P}2 \to 2$ with $h^{\mathcal{P}}(\varphi) = \varphi(1)$, we derive a canonical distributive law $\lambda^{\mathcal{P}}$ between F and the powerset monad $\mathcal{P}$. The minimal pointed $\lambda^{\mathcal{P}}$-bialgebra for $L = (a + b)^* a$ with its underlying CSL structure is depicted in Figure 4.6a. The partially ordered state

[8] Minimal in the sense that every state is reachable by an element of A^* and no two different states observe the same language.

[9] Minimal in the sense that every state is reachable by an element of $T(A^*)$ and no two different states observe the same language.



(a) (b)

Figure 4.6: (a) The minimal CSL-structured DFA for $L = (a+b)^*a$; (b) The canonical RFSA for $L = (a+b)^*a$

space $L = \{[\emptyset] \leq [\{x\}] \leq [\{y\}]\}$ is necessarily finite, thus satisfies the descending chain condition, which turns the set of join-irreducibles into a size-minimal generator $(J(L), i, d)$ with $i(y) = y$ and $d(x) = \{y \in J(L) \mid y \leq x\}$, cf. Lemma 4.4.0.3. In this case, the join-irreducibles are given by all non-zero states. The $\mathcal{P}$-succinct automaton consequently induced by Proposition 4.4.0.5 is depicted in Figure 4.6b; it can be recognised as the canonical RFSA, cf. e.g. [119].

The soundness of the construction in Figure 4.6a can be verified with the following result, which translates the abstract definition of a free $\lambda^{\mathcal{P}}$-bialgebra to concrete data.

Lemma 4.4.0.10. *Let* $(\mathcal{P}X, \mu_X^{\mathcal{P}}, \langle\overline{\varepsilon}, \overline{\delta}\rangle) := \mathsf{free}^{\lambda^{\mathcal{P}}}((X, \langle\varepsilon, \delta\rangle))$. *Then it holds* $\overline{\varepsilon}(\varphi) = \vee_{y\in\varepsilon^{-1}(1)}\varphi(y)$ *and* $\overline{\delta}_a(\varphi)(x) = \vee_{y\in\delta_a^{-1}(x)}\varphi(y)$.

Proof. The first equality is a consequence of

$$\begin{aligned}
\overline{\varepsilon}(\varphi) &= \pi_1 \circ (h^{\mathcal{P}} \times \mathsf{st}) \circ (\mathcal{P}\pi_1, \mathcal{P}\pi_2) \circ \mathcal{P}(\langle\varepsilon, \delta\rangle)(\varphi) && \text{(Definition of } \overline{\varepsilon}\text{)} \\
&= h^{\mathcal{P}} \circ \mathcal{P}(\varepsilon)(\varphi) && \text{(Definition of } \pi_1\text{)} \\
&= \mathcal{P}(\varepsilon)(\varphi)(1) && \text{(Definition of } h^{\mathcal{P}}\text{)} \\
&= \vee_{y\in\varepsilon^{-1}(1)}\varphi(y) && \text{(Definition of } \mathcal{P}(\varepsilon)\text{)}
\end{aligned}$$

For the second equality we observe

$$\begin{aligned}
\overline{\delta}_a(\varphi)(x) &= \overline{\delta}(\varphi)(a)(x) && \text{(Definition of } \overline{\delta}_a\text{)} \\
&= \pi_2 \circ (h^{\mathcal{P}} \times \mathsf{st}) \circ \langle\mathcal{P}\pi_1, \mathcal{P}\pi_2\rangle \circ \mathcal{P}(\langle\varepsilon, \delta\rangle)(\varphi)(a)(x) && \text{(Definition of } \overline{\delta}\text{)} \\
&= \mathsf{st} \circ \mathcal{P}(\delta)(\varphi)(a)(x) && \text{(Definition of } \pi_2\text{)}
\end{aligned}$$

$$
\begin{aligned}
&= \mathcal{P}(\mathsf{ev}_a)(\mathcal{P}(\delta)(\varphi))(x) && \text{(Definition of } \mathsf{st}\text{)}\\
&= \mathcal{P}(\delta_a)(\varphi)(x) && \text{(Definition of } \delta_a\text{)}\\
&= \vee_{y \in \delta_a^{-1}(x)} \varphi(y) && \text{(Definition of } \mathcal{P}(\delta_a)\text{)}
\end{aligned}
$$

□

To cover the canonical *nominal* RFSA we need the following result which shows that the canonical strength[5] function for the powerset monad $\mathcal{P}$ on the category Set naturally lifts to the nominal powerset monad $\mathcal{P}_{\mathrm{n}}$ (cf. Examples 2.2.0.8) on the category $\mathbb{A}$-Nom of finitely supported nominal $\mathbb{A}$-sets and equivariant functions.

Lemma 4.4.0.11. *The strength function* $\mathsf{st} : \mathcal{P}_{\mathrm{n}}(X^A) \to (\mathcal{P}_{\mathrm{n}}X)^A$ *satisfying* $\mathsf{st}(\Phi)(a) = \mathcal{P}_{\mathrm{n}}(\mathsf{ev}_a)(\Phi)$ *is equivariant.*

Proof. As before, let $\mathsf{Perm}(\mathbb{A})$ be the set of permutations of $\mathbb{A}$, that is, bijective functions $\pi : \mathbb{A} \to \mathbb{A}$. We first observe that for any $a \in \mathbb{A}, x \in X$ and $\pi \in \mathsf{Perm}(\mathbb{A})$ the mapping

$$\{\varphi \in X^{\mathbb{A}} \mid \varphi(a) = x\} \to \{\varphi \in X^{\mathbb{A}} \mid \varphi(\pi^{-1}.a) = \pi^{-1}.x\} \qquad \pi \mapsto \pi^{-1}.\varphi \tag{4.2}$$

defines a bijection with inverse assignment $\varphi \mapsto \pi.\varphi$. Note that the set 2 is equipped with the trivial action. The statement thus follows from

$$
\begin{aligned}
(\pi.\mathsf{st}(\Phi))(a)(x) &= \pi.(\mathsf{st}(\Phi)(\pi^{-1}.a))(x) && \text{(Definition of } \pi.\mathsf{st}(\Phi)\text{)}\\
&= \mathsf{st}(\Phi)(\pi^{-1}.a)(\pi^{-1}.x) && \text{(Definition of } \pi.(\mathsf{st}(\Phi)(\pi^{-1}.a)))\\
&= \mathcal{P}_{\mathrm{n}}(\mathsf{ev}_{\pi^{-1}.a})(\Phi)(\pi^{-1}.x) && \text{(Definition of } \mathsf{st}\text{)}\\
&= \vee_{\varphi \in \mathsf{ev}^{-1}_{\pi^{-1}.a}(\pi^{-1}.x)} \Phi(\varphi) && \text{(Definition of } \mathcal{P}_{\mathrm{n}}(\mathsf{ev}_{\pi^{-1}.a})\text{)}\\
&= \vee_{\varphi \in \mathsf{ev}_a^{-1}(x)} \Phi(\pi^{-1}.\varphi) && (4.2)\\
&= \vee_{\varphi \in \mathsf{ev}_a^{-1}(x)} (\pi.\Phi)(\varphi) && \text{(Definition of } \pi.\Phi\text{)}\\
&= \mathcal{P}_{\mathrm{n}}(\mathsf{ev}_a)(\pi.\Phi)(x) && \text{(Definition of } \mathcal{P}_{\mathrm{n}}(\mathsf{ev}_a)\text{)}\\
&= \mathsf{st}(\pi.\Phi)(a)(x) && \text{(Definition of } \mathsf{st}\text{)}
\end{aligned}
$$

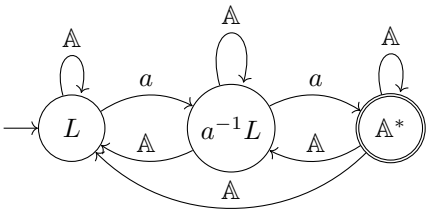
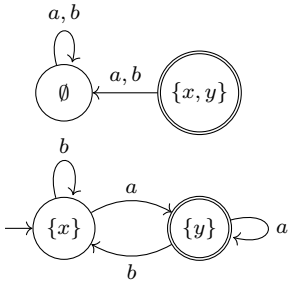


Figure 4.7: The orbit-finite representation of the canonical nominal RFSA for $L = \{vawau \mid v, w, u \in \mathbb{A}^*, a \in \mathbb{A}\}$

□

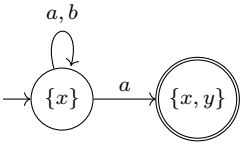
Example 4.4.0.12 (The Canonical Nominal RFSA)**.** Let $\mathbb{A}$ be a countably infinite set. In Lemma 4.4.0.11 we have established that the canonical strength function in Set lifts to $\mathbb{A}$-Nom. It is also not hard to see that the functor F with $FX = 2 \times X^{\mathbb{A}}$ extends to a functor on $\mathbb{A}$-Nom, and $h^{\mathcal{P}_\mathrm{n}} : \mathcal{P}_\mathrm{n} 2 \to 2$ with $h^{\mathcal{P}_\mathrm{n}}(\varphi) = \varphi(1)$ defines a $\mathcal{P}_\mathrm{n}$-algebra, which induces a canonical distributive law $\lambda^{\mathcal{P}_\mathrm{n}}$ between F and the nominal powerset monad $\mathcal{P}_\mathrm{n}$ (cf. Examples 2.2.0.8). As in [111], let $L = \{vawau \mid v, w, u \in \mathbb{A}^*, a \in \mathbb{A}\}$, then $a^{-n}L = a^{-2}L = \mathbb{A}^*$ for $n \geq 2$, and $v^{-1}L = \cup_{a \in \mathbb{A}} a^{-|v|_a} L$, where $|v|_a$ denotes the number of a's that occur in v. In consequence, the nominal CSL underlying the minimal pointed $\lambda^{\mathcal{P}_\mathrm{n}}$-bialgebra is generated by the orbit-finite nominal set of join-irreducibles $\{L\} \cup \{a^{-1}L \mid a \in \mathbb{A}\} \cup \{\mathbb{A}^*\}$, which is equipped with the obvious $\mathsf{Perm}(\mathbb{A})$-action and satisfies the inclusion $L \subseteq a^{-1}L \subseteq \mathbb{A}^*$. The orbit-finite representation of the $\mathcal{P}_\mathrm{n}$-succinct automaton induced by Proposition 4.4.0.5 is depicted in Figure 4.7.

Example 4.4.0.13 (The Minimal Xor Automaton)**.** The $\mathcal{R}$-algebra structure $h^{\mathcal{R}} : \mathcal{R}2 \to 2$ with $h^{\mathcal{R}}(\varphi) = \varphi(1)$ induces a canonical distributive law $\lambda^{\mathcal{R}}$ between F and the free vector space monad $\mathcal{R}$ over the two element field. The minimal pointed $\lambda^{\mathcal{R}}$-bialgebra accepting $L = (a + b)^* a$ is depicted in Figure 4.8a and coincides with the bialgebra freely generated by the F-coalgebra in Figure 4.3. The underlying vector space structure necessarily has a basis. We choose (Y, i, d) with $Y = \{\{x\}, \{x, y\}\}$, $i(y) = y$, and $d(\emptyset) = \emptyset$, $d(\{x\}) = \{\{x\}\}$, $d(\{y\}) = \{\{x\}, \{x, y\}\}$, $d(\{x, y\}) = \{\{x, y\}\}$. The $\mathcal{R}$-succinct automaton induced by Proposition 4.4.0.5 is depicted in Figure 4.8b; it can be recognised as the minimal xor automaton, cf. e.g. [119].



$\oplus$	$\{x\}$	$\{y\}$	$\{x,y\}$	$\emptyset$
$\{x\}$	$\emptyset$	$\{x,y\}$	$\{y\}$	$\{x\}$
$\{y\}$	$\{x,y\}$	$\emptyset$	$\{x\}$	$\{y\}$
$\{x,y\}$	$\{y\}$	$\{x\}$	$\emptyset$	$\{x,y\}$
$\emptyset$	$\{x\}$	$\{y\}$	$\{x,y\}$	$\emptyset$

(a)



(b)

Figure 4.8: (a) The minimal $\mathbb{Z}_2$-vector space structured DFA for $L = (a+b)^*a$ (freely-generated by the DFA in Figure 4.3); (b) Up to the choice of a basis, the minimal xor automaton for $L = (a + b)^*a$

As before, the soundness of the construction in Figure 4.8a can be verified with the help of the following result, which translates the abstract definition of a free $\lambda^{\mathcal{R}}$-bialgebra to concrete data.

Lemma 4.4.0.14. *Let* $(\mathcal{R}X, \mu_X^{\mathcal{R}}, \langle\overline{\varepsilon}, \overline{\delta}\rangle) := \mathsf{free}^{\lambda^{\mathcal{R}}}((X, \langle\varepsilon, \delta\rangle))$. *Then it holds* $\overline{\varepsilon}(\varphi) = \bigoplus_{y\in\varepsilon^{-1}(1)} \varphi(y)$ *and* $\overline{\delta}_a(\varphi)(x) = \bigoplus_{y\in\delta_a^{-1}(x)} \varphi(y)$.

Proof. Analogous to the proof of Lemma 4.4.0.10. □

4.5 Changing the Type of Succinct Automata

This section contains a generalisation of the approach in Section 4.4. The extension is based on the observation that in the last section we implicitly considered *two* types of monads: (i) a monad S that describes the additional algebraic structure of a given deterministic automaton; and (ii) a monad T that captures the side-effects of the succinct automaton that is obtained by the generator-based translation. In Proposition 4.4.0.5, the main result of the last section, the monads coincided, but to recover for instance the átomaton [39] we will have to extend Proposition 4.4.0.5 to a situation where S and T can differ.

4.5.1 Relating Distributive Laws

We now recall the main technical ingredient of our extension: *distributive law homomorphisms*. As before, we present the theory on the level of arbitrary bialgebras, even though we will later focus on the case where the coalgebraic dynamics are those of deterministic automata. Distributive law homomorphisms will allow us to shift a bialgebra over a monad S to an equivalent bialgebra over a monad T, for which we can then find, analogous to Section 4.4, an equivalent succinct representation. The notion we use is an instance of a much more general definition that allows to relate distributive laws on two different categories. We restrict to the case where both distributive laws are given over the same behavioural endofunctor F.

Definition 4.5.1.1 (Distributive Law Homomorphism [157, 128])**.** Let $\lambda^S : SF \to FS$ and $\lambda^T : TF \to FT$ be distributive laws between monads S and T and an endofunctor F, respectively. A *distributive law homomorphism* $\alpha : \lambda^S \to \lambda^T$ consists of a natural transformation $\alpha : T \Rightarrow S$ such that the following three diagrams commute:

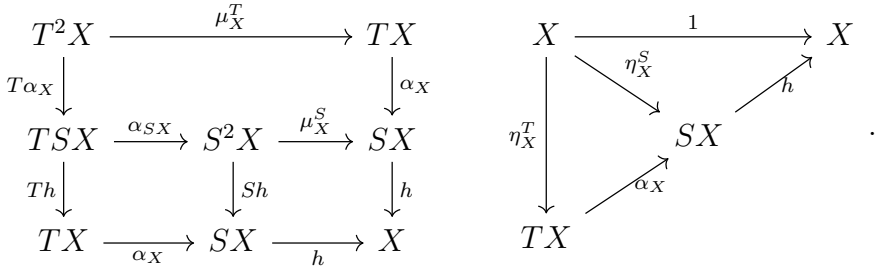
$$\begin{array}{ccc} T^2 & \xrightarrow{\mu^T} & T \\ {\scriptstyle T\alpha}\downarrow & & \downarrow{\scriptstyle \alpha} \\ TS & \xrightarrow[\alpha_S]{} SS \xrightarrow[\mu^S]{} & S \end{array} \qquad \begin{array}{ccc} 1 & \xrightarrow{\eta^S} & S \\ {\scriptstyle \eta^T}\downarrow & \nearrow_{\alpha} & \\ T & & \end{array} \qquad \begin{array}{ccc} TF & \xrightarrow{\lambda^T} & FT \\ {\scriptstyle \alpha_F}\downarrow & & \downarrow{\scriptstyle F\alpha} \\ SF & \xrightarrow[\lambda^S]{} & FS \end{array}.$$

The above definition is such that α induces a functor between the categories of λ^S- and λ^T-bialgebras. The statement is well-known [87, 35]. Since we were unable to locate a full proof in the literature, we include one by ourselves below.

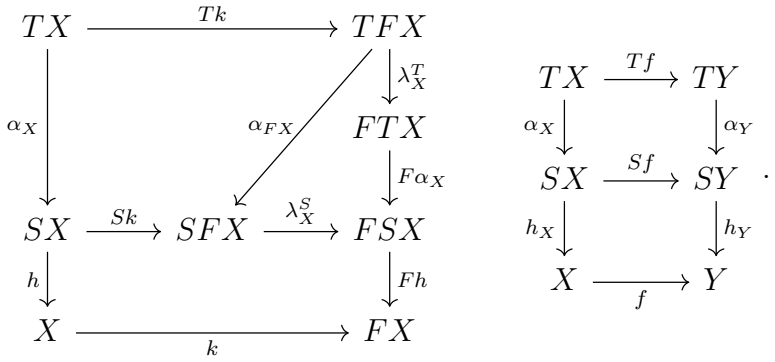
Lemma 4.5.1.2 ([87, 35])**.** *Let $\alpha : \lambda^S \to \lambda^T$ be a distributive law homomorphism. Then $\alpha(X, h, k) := (X, h \circ \alpha_X, k)$ and $\alpha(f) := f$ defines a functor $\alpha : \mathsf{Bialg}(\lambda^S) \to \mathsf{Bialg}(\lambda^T)$.*

Proof. We first show that the construction is well-defined on objects. The commuta-

tivity of the two diagrams below shows that $(X, h \circ \alpha_X)$ is a T-algebra:



To establish that $(X, h \circ \alpha_X, k)$ is a λ^T-bialgebra it thus remains to observe the commutativity of the diagram on the left below:



Well-definedness on morphisms follows from the naturality of α, as seen on the right above. Compositionality follows immediately from the definition of α on morphisms. □

The next result is a straightforward consequence of Proposition 4.4.0.5, and may be strengthened to an isomorphism in case one is given a basis instead of a generator, analogous to Proposition 4.4.0.8. It can be seen as a road map to the approach we propose in this section.

Corollary 4.5.1.3. *Let $\alpha : \lambda^S \to \lambda^T$ be a homomorphism between distributive laws and (X, h, k) a λ^S-bialgebra. If (Y, i, d) is a generator for the T-algebra $(X, h \circ \alpha_X)$, then $(h \circ \alpha_X) \circ Ti : \mathsf{exp}_T((Y, Fd \circ k \circ i)) \to (X, h \circ \alpha_X, k)$ is a λ^T-bialgebra homomorphism.*

Proof. By Lemma 4.5.1.2 the tuple $(X, h \circ \alpha_X, k)$ constitutes a λ^T-bialgebra. The

statement thus follows from Proposition 4.4.0.5. □

4.5.2 Deriving Distributive Law Relations

We now turn to the procedure of deriving a distributive law homomorphism. In practice, coming up with a natural transformation and proving that it lifts to a distributive law homomorphism can be quite cumbersome.

Fortunately, for certain cases, there is a way to simplify things significantly. For instance, as the next result shows, if, as in (4.1), the involved distributive laws are induced by algebra structures h^S and h^T for an output set B, respectively, then one of the conditions is implied by a less convoluted constraint.

Lemma 4.5.2.1. *Let $\alpha : T \Rightarrow S$ be a natural transformation satisfying $h^S \circ \alpha_B = h^T$, then $\lambda^S \circ \alpha_F = F\alpha \circ \lambda^T$.*

Proof. We need to establish the commutativity of the following diagram:

$$\begin{array}{ccc} T(X^A \times B) & \xrightarrow{\alpha_{X^A \times B}} & S(X^A \times B) \\ {\scriptstyle (T\pi_1, T\pi_2)}\downarrow & & \downarrow{\scriptstyle (S\pi_1, S\pi_2)} \\ T(X^A) \times TB & \xrightarrow{\alpha_{X^A} \times \alpha_B} & S(X^A) \times SB \\ {\scriptstyle \mathsf{st} \times h^T}\downarrow & & \downarrow{\scriptstyle \mathsf{st} \times h^S} \\ (TX)^A \times B & \xrightarrow[(\alpha_X)^A \times B]{} & (SX)^A \times B \end{array} .$$

The commutativity of the top square is a consequence of the naturality of α. Similarly, the commutativity of the bottom square follows from the assumption and the naturality of α,

$$\begin{aligned} \mathsf{st} \circ \alpha_{X^A}(U)(a) &= S(\mathsf{ev}_a) \circ \alpha_{X^A}(U) && \text{(Definition of } \mathsf{st}) \\ &= \alpha_X \circ T(\mathsf{ev}_a)(U) && \text{(Naturality of } \alpha) \\ &= \alpha_X^A \circ \mathsf{st}(U)(a) && \text{(Definition of } \mathsf{st}) \end{aligned}$$

□

The next result shows that for the neighbourhood monad there exists a family of *canonical* choices of distributive law homomorphisms parametrised by Eilenberg-Moore algebra structures on the output set $B = 2$. While it is well-known that such algebras induce a monad morphism, for instance in the coalgebraic modal logic community [85, 136, 62], its commutativity with canonical distributive laws has not been observed before. Moreover, we provide a new formalisation in terms of the strength function, which allows the result to be lifted to strong monads and arbitrary output objects on other categories than the one of sets and functions.

Proposition 4.5.2.2. *Any algebra $h : T2 \to 2$ over a set monad T induces a homomorphism $\alpha^h : \lambda^{\mathcal{H}} \to \lambda^h$ between distributive laws by $\alpha^h_X := h^{2^X} \circ \mathsf{st} \circ T(\eta^{\mathcal{H}}_X)$.*

Proof. It is well-known that the strength operation is natural and satisfies the two equalities

$$(\eta^T_A)^B = \mathsf{st} \circ \eta^T_{A^B} \qquad \mathsf{st} \circ \mu_{A^B} = \mu^B_A \circ \mathsf{st} \circ T\mathsf{st}.$$

It is also not hard to see that for functions $f : A \to B$ and $g : C \to D$ it holds $f^D \circ A^g = B^g \circ f^C$. We write f^* for A^f and f_* for f^A, and omit components of natural transformations for readability. The naturality of α^{h^T} is a consequence of:

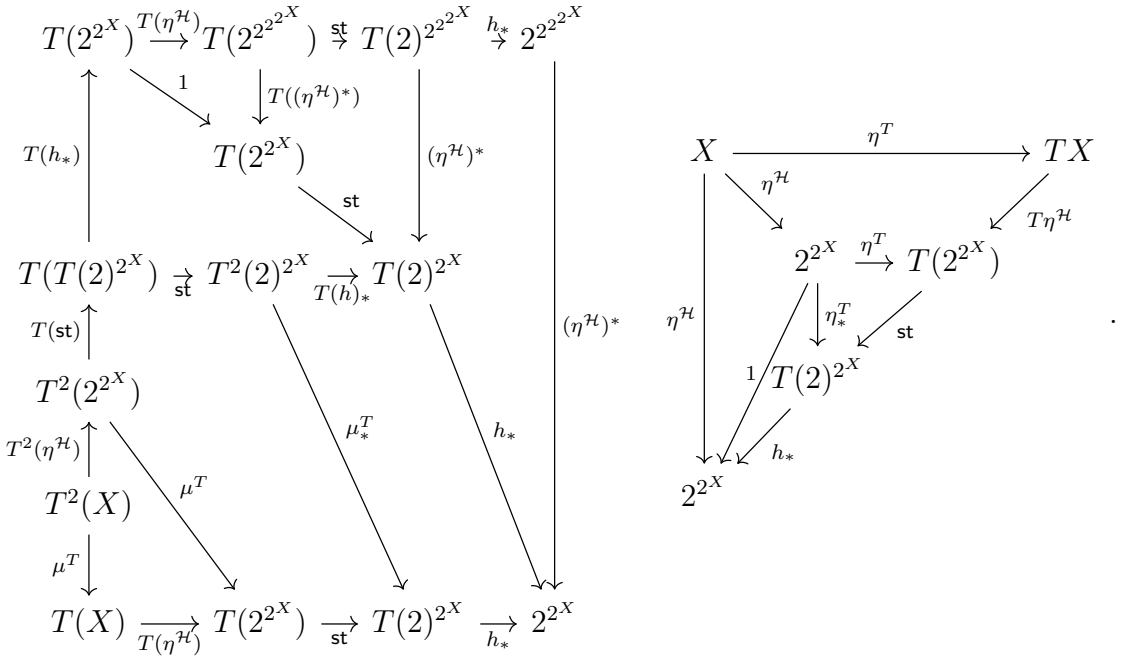
$$\begin{array}{ccccccc} TX & \xrightarrow{T\eta^{\mathcal{H}}} & T(2^{2^X}) & \xrightarrow{\mathsf{st}} & T(2)^{2^X} & \xrightarrow{h_*} & 2^{2^X} \\ {\scriptstyle Tf}\downarrow & & \downarrow{\scriptstyle T((f^*)^*)} & & \downarrow{\scriptstyle (f^*)^*} & & \downarrow{\scriptstyle (f^*)^*} \\ TY & \xrightarrow[T\eta^{\mathcal{H}}]{} & T(2^{2^Y}) & \xrightarrow[\mathsf{st}]{} & T(2)^{2^Y} & \xrightarrow[h_*]{} & 2^{2^Y} \end{array}.$$

Next, note that $2^{\eta^{\mathcal{H}}_{2^X}} \circ \eta^{\mathcal{H}}_{2^{2^X}} = \mathsf{id}_{2^{2^X}}$, as for all $\Phi \in 2^{2^X}, \varphi \in 2^X$ we have:

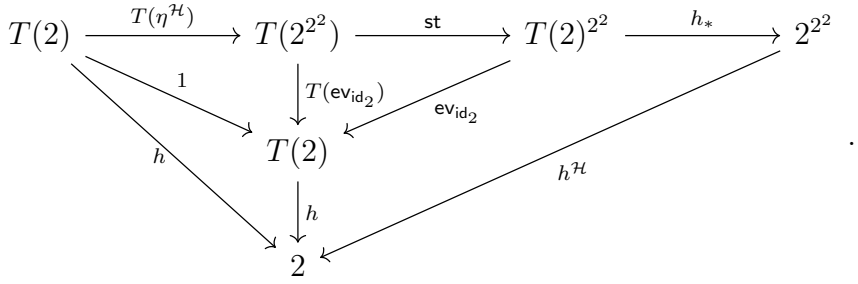
$$\begin{aligned} (2^{\eta^{\mathcal{H}}_{2^X}} \circ \eta^{\mathcal{H}}_{2^{2^X}})(\Phi)(\varphi) &= (\eta^{\mathcal{H}}_{2^{2^X}}(\Phi) \circ \eta^{\mathcal{H}}_{2^X})(\varphi) && \text{(Definition of } 2^f\text{)} \\ &= \eta^{\mathcal{H}}_{2^X}(\varphi)(\Phi) && \text{(Definition of } \eta^{\mathcal{H}}_{2^{2^X}}\text{)} \\ &= \Phi(\varphi) && \text{(Definition of } \eta^{\mathcal{H}}_{2^X}\text{)} \\ &= \mathsf{id}_{2^{2^X}}(\Phi)(\varphi) && \text{(Definition of } \mathsf{id}_{2^{2^X}}\text{).} \end{aligned}$$

Using the above equality, the equation involving the monad multiplications is seen from the diagram on the left below. The equation involving the monad units is

established by the diagram on the right below:



To show that the equation involving the distributive laws holds, we use Lemma 4.5.2.1. That is, we note that for any f it holds $f \circ \mathsf{ev}_a = \mathsf{ev}_a \circ f_*$ and $h^{\mathcal{H}} = \mathsf{ev}_{\mathsf{id}_2}$, before establishing the following commutative diagram:



□

The rest of the section is concerned with using Proposition 4.5.2.2 and Corollary 4.5.1.3 to derive canonical acceptors through distributive law homomorphisms.

4.5.3 Example: The Átomaton

We will now justify the previous informal construction of the átomaton. As hinted before, the átomaton can be recovered by relating the neighbourhood monad $\mathcal{H}$—whose algebras are complete *atomic* Boolean algebras (CABAs)—and the powerset monad $\mathcal{P}$. Formally, as a consequence of Proposition 4.5.2.2 we obtain the following.

Corollary 4.5.3.1. *Let $\alpha_X : \mathcal{P}X \to \mathcal{H}X$ satisfy $\alpha_X(\varphi)(\psi) = \vee_{x\in X}\varphi(x) \wedge \psi(x)$, then α constitutes a distributive law homomorphism $\alpha : \lambda^{\mathcal{H}} \to \lambda^{\mathcal{P}}$.*

Proof. We show that $\alpha^{h^{\mathcal{P}}} = \alpha$, the statement then follows from Proposition 4.5.2.2:

$$\begin{aligned}
\alpha_X^{h^{\mathcal{P}}}(\varphi)(\psi) &= (h^{\mathcal{P}})^{2^X} \circ \mathsf{st} \circ \mathcal{P}(\eta_X^{\mathcal{H}})(\varphi)(\psi) && \text{(Definition of } \alpha_X^{h^{\mathcal{P}}}\text{)}\\
&= \mathsf{st} \circ \mathcal{P}(\eta_X^{\mathcal{H}})(\varphi)(\psi)(1) && \text{(Definition of } h^{\mathcal{P}}\text{)}\\
&= \mathcal{P}(\mathsf{ev}_\psi)(\mathcal{P}(\eta_X^{\mathcal{H}})(\varphi))(1) && \text{(Definition of } \mathsf{st}\text{)}\\
&= \mathcal{P}(\mathsf{ev}_\psi \circ \eta_X^{\mathcal{H}})(\varphi)(1) && (\mathcal{P}(f) \circ \mathcal{P}(g) = \mathcal{P}(f \circ g))\\
&= \mathcal{P}(\psi)(\varphi)(1) && \text{(Definition of } \mathsf{ev}, \eta_X^{\mathcal{H}}\text{)}\\
&= \vee_{x\in\psi^{-1}(1)}\varphi(x) && \text{(Definition of } \mathcal{P}(\psi)\text{)}\\
&= \vee_{x\in X}\varphi(x) \wedge \psi(x) && (x \in \psi^{-1}(1))\\
&= \alpha_X(\varphi)(\psi) && \text{(Definition of } \alpha_X\text{)}
\end{aligned}$$

□

The next statement follows from a well-known Stone-type duality representation theorem for CABAs [149].

Lemma 4.5.3.2. *Let $\alpha_X : \mathcal{P}X \to \mathcal{H}X$ satisfy $\alpha_X(\varphi)(\psi) = \vee_{x\in X}\varphi(x) \wedge \psi(x)$. If $B = (X, h)$ is an $\mathcal{H}$-algebra, then $(\mathsf{At}(B), i, d)$ with $i(a) = a$ and $d(x) = \{a \in \mathsf{At}(B) \mid a \leq x\}$ is a basis for the $\mathcal{P}$-algebra $(X, h \circ \alpha_X)$.*

Proof. Let $K : \mathsf{Set}^{\mathrm{op}} \to \mathsf{Set}^{\mathcal{H}}$ denote the comparison functor with $K(X) = (\mathcal{P}X, 2^{\eta_X^{\mathcal{H}}})$ induced by the self-dual contravariant powerset adjunction. It is well-known that K has a quasi-inverse, namely the functor $\mathsf{At} : \mathsf{Set}^{\mathcal{H}} \to \mathsf{Set}^{\mathrm{op}}$ assigning to a complete atomic Boolean algebra B its atoms $\mathsf{At}(B)$ [149]. The equivalence $d : B \simeq K \circ \mathsf{At}(B)$

is given by $d(x) = \{a \in \mathsf{At}(B) \mid a \leq x\}$. The calculation below

$$\begin{aligned}
2^{\eta^{\mathcal{H}}_X} \circ \alpha_{\mathcal{P}X}(\Phi)(x) &= \alpha_{\mathcal{P}X}(\Phi)(\eta^{\mathcal{H}}_X(x)) && \text{(Definition of } 2^{\eta^{\mathcal{H}}_X}) \\
&= \vee_{\varphi \in 2^X} \Phi(\varphi) \wedge \eta^{\mathcal{H}}_X(x)(\varphi) && \text{(Definition of } \alpha_{\mathcal{P}X}) \\
&= \vee_{\varphi \in 2^X} \Phi(\varphi) \wedge \varphi(x) && \text{(Definition of } \eta^{\mathcal{H}}_X) \\
&= \mu^{\mathcal{P}}_X(\Phi)(x) && \text{(Definition of } \mu^{\mathcal{P}}_X)
\end{aligned}$$

shows that $2^{\eta^{\mathcal{H}}_X} \circ \alpha_{\mathcal{P}X} = \mu^{\mathcal{P}}_X$. By Corollary 4.5.3.1 the definition $\alpha((X,h)) = (X, h \circ \alpha_X)$ yields a functor $\alpha : \mathsf{Set}^{\mathcal{H}} \to \mathsf{Set}^{\mathcal{P}}$. We can thus deduce the following equivalence between $\mathcal{P}$-algebras:

$$\begin{aligned}
(X, h \circ \alpha_X) &= \alpha(B) && \text{(Definition of } \alpha) \\
&\simeq \alpha \circ K \circ \mathsf{At}(B) && (\mathsf{id} \simeq K \circ \mathsf{At}) \\
&= (\mathcal{P}(\mathsf{At}(B)), 2^{\eta^{\mathcal{H}}_{\mathsf{At}(B)}} \circ \alpha_{\mathcal{P}(\mathsf{At}(B))}) && \text{(Definition of } \alpha \circ K \circ \mathsf{At}) \\
&= (\mathcal{P}(\mathsf{At}(B)), \mu^{\mathcal{P}}_{\mathsf{At}(B)}) && (2^{\eta^{\mathcal{H}}_X} \circ \alpha_{\mathcal{P}X} = \mu^{\mathcal{P}}_X)
\end{aligned}$$

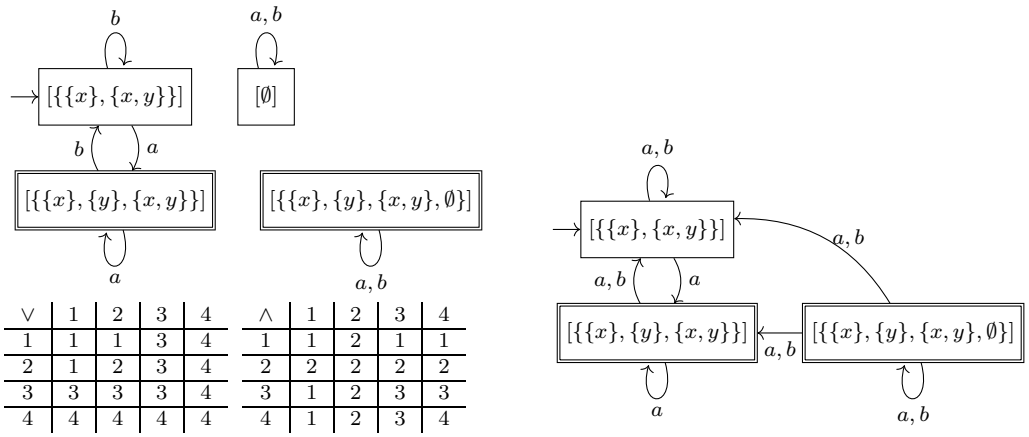
Using the definition of a basis, the former immediately implies the claim. □

The átomaton for the regular language $L = (a+b)^*a$, for example, can now be obtained as follows. First, we construct the minimal pointed $\lambda^{\mathcal{H}}$-bialgebra accepting L, which is depicted in Figure 4.4 together with its underlying CABA structure B. The construction can be verified with the help of the following result.

Lemma 4.5.3.3. *Let* $(\mathcal{H}X, \mu^{\mathcal{H}}_X, \langle \overline{\varepsilon}, \overline{\delta} \rangle) := \mathsf{free}^{\lambda^{\mathcal{H}}}((X, \langle \varepsilon, \delta \rangle))$. *Then it holds* $\overline{\varepsilon}(\Phi) = \Phi(\varepsilon)$ *and* $\overline{\delta}_a(\Phi)(\varphi) = \Phi(\varphi \circ \delta_a)$.

Proof. The proof is analogous to the one of Lemma 4.4.0.10. The first equality is seen as follows:

$$\begin{aligned}
\overline{\varepsilon}(\Phi) &= \mathcal{H}(\varepsilon)(\Phi)(\mathsf{id}_2) && \text{(Cf. proof of Lemma 4.4.0.10)} \\
&= \Phi(\mathsf{id}_2 \circ \varepsilon) && \text{(Definition of } \mathcal{H}(\varepsilon)) \\
&= \Phi(\varepsilon) && (\mathsf{id}_2 \circ \varepsilon = \varepsilon)
\end{aligned}$$



∨	1	2	3	4
1	1	1	3	4
2	1	2	3	4
3	3	3	3	4
4	4	4	4	4

∧	1	2	3	4
1	1	2	1	1
2	2	2	2	2
3	1	2	3	3
4	1	2	3	4

(a) (b)

Figure 4.9: (a) The minimal CDL-structured DFA for $L = (a + b)^*a$, where $1 \equiv [\{\{x\}, \{x, y\}\}]$, $2 \equiv [\emptyset]$, $3 \equiv [\{\{x\}, \{y\}, \{x, y\}\}]$, $4 \equiv [\{\{x\}, \{y\}, \{x, y\}, \emptyset\}]$; (b) The distromaton for $L = (a + b)^*a$

For the second equality we observe:

$$\begin{aligned} \overline{\delta}_a(\Phi)(\varphi) &= \mathcal{H}(\delta_a)(\Phi)(\varphi) && \text{(Cf. proof of Lemma 4.4.0.10)} \\ &= \Phi(\varphi \circ \delta_a) && \text{(Definition of } \mathcal{H}(\delta_a)) \end{aligned}$$

□

Using the distributive law homomorphism α of Corollary 4.5.3.1, the minimal pointed $\lambda^{\mathcal{H}}$-bialgebra can then be translated into an equivalent pointed $\lambda^{\mathcal{P}}$-bialgebra with underlying CSL-structure $\alpha(B)$. By Lemma 4.5.3.2 the atoms $\mathsf{At}(B)$ of B form a basis for $\alpha(B)$. In this case the atoms are given by $[\{\{x\}, \{x, y\}\}]$, $[\{\{y\}\}]$ and $[\{\emptyset\}]$. The $\mathcal{P}$-succinct automaton consequently induced by Corollary 4.5.1.3 is depicted in Figure 4.5; it can be recognised as the átomaton, cf. e.g. [119].

4.5.4 Example: The Distromaton

We shall now use our framework to recover another canonical non-deterministic acceptor: the *distromaton* [119]. As the name suggests, it can be constructed by relating the monotone neighbourhood monad $\mathcal{A}$ – whose algebras are completely *distributive*

lattices – and the powerset monad $\mathcal{P}$. Formally, the relationship can be established by the same natural transformation we used for the átomaton.

Corollary 4.5.4.1. *Let $\alpha_X : \mathcal{P}X \to \mathcal{A}X$ satisfy $\alpha_X(\varphi)(\psi) = \vee_{x \in X} \varphi(x) \wedge \psi(x)$, then α constitutes a distributive law homomorphism $\alpha : \lambda^{\mathcal{A}} \to \lambda^{\mathcal{P}}$.*

Proof. We observe that $\alpha_X(\varphi) : (2^X, \subseteq) \to (2, \leq)$ is monotone for all $\varphi \in 2^X$. Since the monotone neighbourhood monad $\mathcal{A}$ and the neighbourhood monad $\mathcal{H}$ only differ on objects, the result follows from Corollary 4.5.3.1. □

The distromaton for the regular language $L = (a + b)^* a$, for example, can now be obtained as follows. First, we construct the minimal pointed $\lambda^{\mathcal{A}}$-bialgebra for L, depicted in Figure 4.9a with its underlying CDL structure h. The construction can be verified with the help of the result below.

Lemma 4.5.4.2. *Let $(\mathcal{A}X, \mu^{\mathcal{A}}_X, \langle \overline{\varepsilon}, \overline{\delta} \rangle) := \mathsf{free}^{\lambda^{\mathcal{A}}}((X, \langle \varepsilon, \delta \rangle))$. Then it holds $\overline{\varepsilon}(\Phi) = \Phi(\varepsilon)$ and $\overline{\delta}_a(\Phi)(\varphi) = \Phi(\varphi \circ \delta_a)$.*

Proof. Analogous to the proof of Lemma 4.5.3.3. □

Using the distributive law homomorphism α in Corollary 4.5.4.1, the minimal pointed $\lambda^{\mathcal{A}}$-bialgebra can be translated into an equivalent pointed $\lambda^{\mathcal{P}}$-bialgebra with underlying CSL structure $L = h \circ \alpha_X$. Its partially ordered state space

$$[\emptyset] \leq [\{\{x\}, \{x, y\}\}] \leq [\{\{x\}, \{y\}, \{x, y\}\}] \leq [\{\{x\}, \{y\}, \{x, y\}, \emptyset\}]$$

is necessarily finite, which turns the set of join-irreducibles into a size-minimal generator $(J(L), i, d)$ for L, where $i(y) = y$ and $d(x) = \{y \in J(L) \mid y \leq x\}$. In this case, the join-irreducibles are given by all non-zero states. The $\mathcal{P}$-succinct automaton consequently induced by Corollary 4.5.1.3 is depicted in Figure 4.9b and can be recognised as the distromaton, cf. [119].

4.5.5 Example: The Minimal Xor-CABA Automaton

We conclude this section by relating the neighbourhood monad $\mathcal{H}$ with the free vector space monad $\mathcal{R}$ over the unique two element field $\mathbb{Z}_2$. In particular, we derive a new

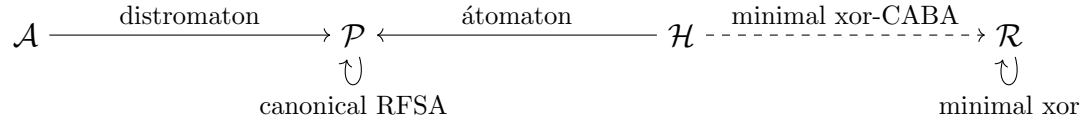


Figure 4.10: The minimal xor-CABA automaton is to the minimal xor automaton what the átomaton is to the canonical RFSA

canonical succinct acceptor for regular languages, which we call the *minimal xor-CABA automaton*.

The next result shows that every CABA can be equipped with a symmetric difference operation that turns it into a vector space over the two element field.

Corollary 4.5.5.1. *Let $\alpha_X : \mathcal{R}X \to \mathcal{H}X$ satisfy $\alpha_X(\varphi)(\psi) = \bigoplus_{x\in X} \varphi(x)\cdot\psi(x)$, then α constitutes a distributive law homomorphism $\alpha : \lambda^{\mathcal{H}} \to \lambda^{\mathcal{R}}$.*

Proof. Analogous to the proof of Corollary 4.5.3.1. □

Since every vector space admits a basis, the above result leads to the definition of a new acceptor of regular languages. In what follows, let α denote the homomorphism in Corollary 4.5.5.1 and F the endofunctor given by $FX = 2 \times X^A$.

Definition 4.5.5.2 (Minimal Xor-CABA Automaton)**.** Let (X, h, k) be the minimal x-pointed $\lambda^{\mathcal{H}}$-bialgebra accepting a regular language $L \subseteq A^*$, and $B = (Y, i, d)$ a basis for the $\mathcal{R}$-algebra $(X, h \circ \alpha_X)$. The *minimal xor-CABA automaton* for L with respect to B is the $d(x)$-pointed $\mathbb{Z}_2$-weighted automaton $Fd \circ k \circ i$.

In Figure 4.10 it is indicated how the canonical acceptors in this chapter, including the minimal xor-CABA automaton, are based on relations between pairs of monads.

For the regular language $L = (a+b)^*a$ the above definition instantiates as follows. First, as for the átomaton, we construct the minimal pointed $\lambda^{\mathcal{H}}$-bialgebra (X, h, k) for L; it is depicted in Figure 4.4. As one easily verifies, the $\mathbb{Z}_2$-vector space $(X, h\circ\alpha_X)$ is induced by the symmetric difference operation $\oplus$ on subsets. Using the notation in Figure 4.4, we choose the basis (Y, i, d) with $Y = \{4, 6, 7, 8\}$; $i(y) = y$; and $d(1) = 7 \oplus 8$, $d(2) = \emptyset$, $d(3) = 6 \oplus 7$, $d(4) = 4$, $d(5) = 6 \oplus 7 \oplus 8$, $d(6) = 6$, $d(7) = 7$,

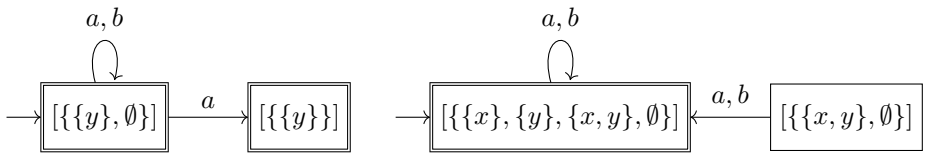


Figure 4.11: The minimal xor-CABA automaton for $L = (a + b)^* a$

$d(8) = 8$. The induced $d(1) = 7 \oplus 8$-pointed $\mathcal{R}$-succinct automaton accepting L, i.e. the minimal xor-CABA automaton, is depicted in Figure 4.11.

4.6 Minimality

In this section we restrict ourselves to the category of (nominal) sets. We show that every language satisfying a suitable property parametric in monads S and T admits a size-minimal succinct automaton of type T accepting it. As a main result we obtain Theorem 4.6.0.5. In Section 4.6.1 we instantiate the former to subsume known minimality results for canonical automata, to prove the xor-CABA automaton minimal, and to establish a size-comparison between different acceptors.

Given a distributive law homomorphism $\alpha : \lambda^S \to \lambda^T$, let $\mathsf{ext} : \mathsf{Coalg}(FT) \to \mathsf{Coalg}(FS)$ be the functor given by $\mathsf{ext}((X, k)) = (X, F\alpha_X \circ k)$ and $\mathsf{ext}(f) = f$. Moreover, let $\mathsf{exp}_U : \mathsf{Coalg}(FU) \to \mathsf{Bialg}(\lambda^U)$ for $U \in \{S, T\}$ denote the functor introduced in Lemma 4.3.0.10.

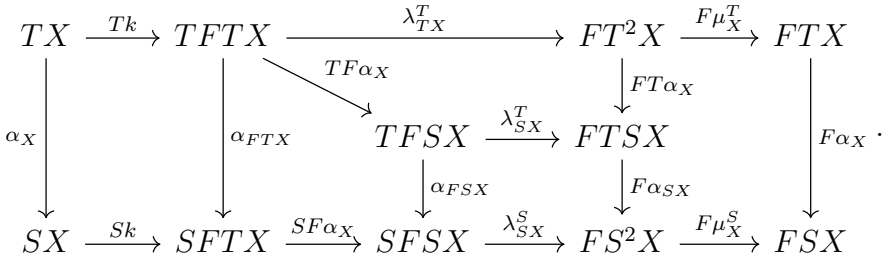
Proposition 4.6.0.1. *Let $\alpha : \lambda^S \to \lambda^T$ be a distributive law homomorphism. Then $\alpha_X : TX \to SX$ underlies a natural transformation $\alpha : \mathsf{exp}_T \Rightarrow \alpha \circ \mathsf{exp}_S \circ \mathsf{ext}$ between functors of type $\mathsf{Coalg}(FT) \to \mathsf{Bialg}(\lambda^T)$.*

Proof. Given a T-succinct automaton $\mathcal{X} = (X, k)$ the definitions imply:

$$\mathsf{exp}_T(\mathcal{X}) = (TX, \mu^T_X, F\mu^T_X \circ \lambda^T_{TX} \circ Tk)$$
$$\alpha \circ \mathsf{exp}_S \circ \mathsf{ext}(\mathcal{X}) = (SX, \mu^S_X \circ \alpha_{SX}, F\mu^S_X \circ \lambda^S_{SX} \circ SF\alpha_X \circ Sk).$$

By the definition of distributive law homomorphisms, the morphism α_X commutes with the underlying T-algebra structures. Its commutativity with the underlying

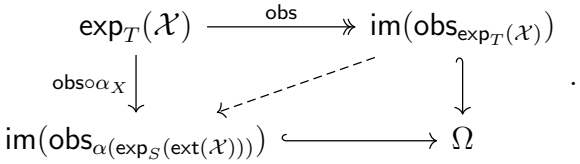
F-coalgebra structures follows from:



Above we use the naturality of α and λ^T, and the definition of a distributivity law homomorphism. The naturality of α as natural transformation $\alpha : \mathsf{exp}_T \Rightarrow \alpha \circ \mathsf{exp}_S \circ \mathsf{ext}$ follows from the naturality of α as natural transformation $\alpha : T \Rightarrow S$. □

In the above situation a T-succinct automaton admits *two* semantics, by lifting the former either to a bialgebra over λ^S or λ^T. The next definition introduces a notion of *closedness* that captures the cases in which the image of both semantics coincides.

Definition 4.6.0.2 (α-Closed Succinct Automaton)**.** Let $\alpha : \lambda^S \to \lambda^T$ be a distributive law homomorphism. We say that a T-succinct automaton $\mathcal{X}$ is *α-closed* if the unique diagonal below is an isomorphism:



It immediately follows that the unique diagonal in Definition 4.6.0.2 is injective.

The result below shows that instead of defining the α-closedness of $\mathcal{X}$ as true if the unique diagonal is an isomorphism, we could have equivalently defined it to be true if their exists *any* isomorphism of such type.

Lemma 4.6.0.3. *$\mathcal{X}$ is α-closed iff* $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \cong \mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))})$.

Proof. Clearly, if $\mathcal{X}$ is α-closed, then, by definition, the unique diagonal in Definition 4.6.0.2 witnesses the desired isomorphism.

Conversely, assume there is *any* isomorphism $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \overset{\varphi}{\cong} \mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))})$. As the unique diagonal d in Definition 4.6.0.2 is necessarily injective, it remains to

show that it is also surjective, that is $\mathsf{im}(d) \cong \mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))})$. The latter follows immediately from the uniqueness of epi-mono factorisations:

$$\begin{array}{ccccc} \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) & & \xrightarrow{\quad d \quad} & & \mathsf{im}(d) \\ \downarrow{\scriptstyle \varphi} & & \overset{\cong}{\leftarrow\!\!-\!\!-\!\!-} & & \downarrow \\ \mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))}) & \xhookrightarrow[\varphi^{-1}]{} & \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) & \xhookrightarrow[d]{} & \mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))}) \end{array} .$$

□

Succinct automata obtained from generators for bialgebras are α-closed.

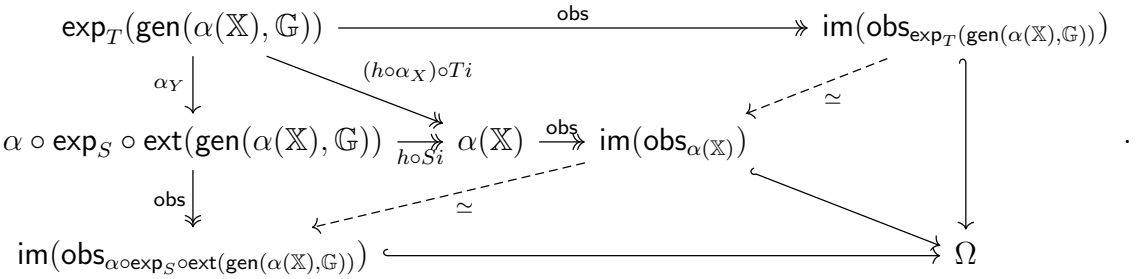
Lemma 4.6.0.4. *Let $\alpha : \lambda^S \to \lambda^T$ be a distributive law homomorphism and (X, h, k) a λ^S-bialgebra. If (Y, i, d) is a generator for $(X, h \circ \alpha_X)$, then $(Y, Fd \circ k \circ i)$ is α-closed.*

Proof. We write $\mathbb{X} := (X, h, k)$, $\mathbb{G} := (Y, i, d)$, and $\mathsf{gen}(\alpha(\mathbb{X}), \mathbb{G}) := (Y, Fd \circ k \circ i)$. The definitions imply

$$\begin{aligned} \mathsf{exp}_T(\mathsf{gen}(\alpha(\mathbb{X}), \mathbb{G})) &= (TY, \mu^T_Y, (Fd \circ k \circ i)^\sharp) \\ \alpha \circ \mathsf{exp}_S \circ \mathsf{ext}(\mathsf{gen}(\alpha(\mathbb{X}), \mathbb{G})) &= (SY, \mu^S_Y \circ \alpha_{SY}, (F(\alpha_Y \circ d) \circ k \circ i)^\sharp). \end{aligned}$$

Since $\mathbb{G}$ is a generator for $(X, h \circ \alpha_X)$, Proposition 4.4.0.5 implies that $(h \circ \alpha_X) \circ Ti : \mathsf{exp}_T(\mathsf{gen}(\alpha(\mathbb{X}), \mathbb{G})) \to \alpha(\mathbb{X})$ is a λ^T-bialgebra homomorphism. By the definition of a generator, $(h \circ \alpha_X) \circ Ti$ has a right-inverse, which implies its surjectivity. Naturality of α shows that $\overline{G} = (Y, i, \alpha_Y \circ d)$ is a generator for (X, h). Thus Proposition 4.4.0.5 implies that $h \circ Si : \mathsf{exp}_S(\mathsf{gen}(\mathbb{X}, \overline{\mathbb{G}})) \to \mathbb{X}$ is a λ^S-bialgebra homomorphism. By the definition of a generator, $h \circ Si$ has a right-inverse, which implies its surjectivity. Applying α shows that $h \circ Si : \alpha \circ \mathsf{exp}_S \circ \mathsf{ext}(\mathsf{gen}(\alpha(\mathbb{X}), \mathbb{G})) \to \alpha(\mathbb{X})$ is a surjective λ^T-bialgebra homomorphism. The statement follows from the uniqueness of epi-mono

factorisations:



□

We are now able to state our main result.

In [119, Theorem 4.8] it is shown that i) the canonical RFSA for L accepts the least number of languages among all NFAs accepting L; and ii) among NFAs with the same number of accepted languages, the canonical RFSA is state-minimal. Note how the two bullet points of Theorem 4.6.0.5 below resemble the ones in [119, Theorem 4.8]: if $S = T = \mathcal{P}$ and α is trivial, the theorem implies that i) the set of languages accepted by the canonical RFSA for L is included in any other set of languages accepted by NFAs for L; and ii) among NFAs for L that accept the same set of languages (i.e. $\overline{\mathsf{Der}(L)}^{\mathsf{CSL}}$), the canonical RFSA is size-minimal (cf. Corollary 4.6.1.3).

Theorem 4.6.0.5 (Minimal Succinct Automata)**.** *Given a language $L \in \Omega$ such that there exists a minimal pointed λ^S-bialgebra $\mathbb{M}$ accepting L and the underlying algebra of $\alpha(\mathbb{M})$ admits a size-minimal generator, there exists a pointed α-closed T-succinct automaton $\mathcal{X}$ accepting L such that:*

- *for any pointed α-closed T-succinct automaton $\mathcal{Y}$ accepting L we have that $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \subseteq \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{Y})})$;*
- *if $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) = \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{Y})})$, then $|X| \leq |Y|$, where X and Y are the carriers of $\mathcal{X}$ and $\mathcal{Y}$, respectively.*

Proof. We use a similar notation as in the proof of Lemma 4.6.0.4. Let $\mathbb{G} = (X, i, d)$ be the size-minimal generator for the underlying algebra of $\alpha(\mathbb{M})$, which we assume to be x-pointed. We define a $d(x)$-pointed T-succinct automaton $\mathcal{X} := \mathsf{gen}(\alpha(\mathbb{M}), \mathbb{G})$.

By Corollary 4.5.1.3 there exists a λ^T-bialgebra homomorphism $i^\sharp \colon \mathsf{exp}_T(\mathcal{X}) \to \alpha(\mathbb{M})$. By the definition of a generator, $i^\sharp$ has a right-inverse, which implies its surjectivity. From the uniqueness of morphisms into a final coalgebra we can deduce that $\mathcal{X}$ accepts the language accepted by $\alpha(\mathbb{M})$. Because α only modifies the algebraic part of a bialgebra and the final bialgebra homomorphism is induced by the underlying final coalgebra homomorphism, the language accepted by $\alpha(\mathbb{M})$ is the language L accepted by $\mathbb{M}$. From Lemma 4.6.0.4 it follows that $\mathcal{X}$ is α-closed.

Consider any pointed α-closed T-succinct automaton $\mathcal{Y}$ accepting L. By minimality of $\mathbb{M}$ there is an injective λ^S-bialgebra homomorphism $j \colon \mathbb{M} \hookrightarrow \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{Y}))})$, which is also an injective λ^T-bialgebra homomorphism $j \colon \alpha(\mathbb{M}) \hookrightarrow \mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{Y})))})$, because the functor α is the identity on morphisms, and only modifies the algebraic part of a bialgebra. Since $\mathcal{Y}$ is α-closed, we have an isomorphism $\mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{Y})))}) \cong \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{Y})})$. By composing the previous functions, we thus obtain an injective λ^T-bialgebra homomorphism $k : \alpha(\mathbb{M}) \hookrightarrow \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{Y})})$. Since any final bialgebra homomorphism is induced by the underlying final F-coalgebra homomorphism, we have $\mathsf{obs}_{\alpha(\mathbb{M})} = \mathsf{obs}_{\mathbb{M}}$. By assumption $\mathbb{M}$ is minimal, that is, $\mathsf{obs}_{\mathbb{M}}$ is injective. By the uniqueness of morphisms into a final object the outer square of the diagram below commutes:

$$\begin{array}{ccc} \mathsf{exp}_T(\mathcal{X}) & \xrightarrow{\;i^\sharp\;} & \alpha(\mathbb{M}) \\ {\scriptstyle \mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}}\downarrow & \swarrow \cong & \downarrow{\scriptstyle \mathsf{obs}_{\alpha(\mathbb{M})} = \mathsf{obs}_{\mathbb{M}}} \\ \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) & \hookrightarrow & \Omega \end{array}$$

By the uniqueness of epi-mono factorisations, there thus exists a diagonal isomorphism $\alpha(\mathbb{M}) \cong \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})})$. By composing the diagonal isomorphism with k, we obtain a λ^T-bialgebra homomorphism $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \to \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{Y})})$. The latter commutes with observability maps and thus is an inclusion, so $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \subseteq \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{Y})})$.

Suppose $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) = \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{Y})})$, which implies that j is an isomorphism. Then there exists a surjective λ^T-bialgebra homomorphism $\mathsf{exp}_T(\mathcal{Y}) \to \alpha(\mathbb{M})$, which means that Y forms the carrier of a generator for the underlying algebra of $\alpha(\mathbb{M})$. By the size-minimality of $\mathbb{G}$ we thus obtain $|X| \leq |Y|$. □

For a T-succinct automaton $\mathcal{X}$ let us write $\mathsf{obs}^\dagger_{\mathcal{X}} := \mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})} \circ \eta^T_X : X \to \Omega$ for a generalisation of the semantics of non-deterministic automata. Lemma 4.6.0.8

provides an equivalent characterisation of α-closedness in terms of $\mathsf{obs}^\dagger$ that will be particularly useful in Section 4.6.1. To prove Lemma 4.6.0.8, we need the following technical results.

Lemma 4.6.0.6. *Let $\alpha : \lambda^S \to \lambda^T$ be a distributive law homomorphism. If (Ω, h, k) is the final λ^S-bialgebra, then $(\Omega, h \circ \alpha_\Omega, k)$ is the final λ^T-bialgebra.*

Proof. It is well-known that if (Ω, h, k) is the final λ^S-bialgebra, then (Ω, k) is the final F-coalgebra and $h : S\Omega \to \Omega$ is the unique homomorphism satisfying $k \circ h = Fh \circ \lambda^S_\Omega \circ Sk$. Similarly, it is well-known that $(\Omega, \overline{h}, k)$ is the final λ^T-bialgebra, where $\overline{h} : T\Omega \to \Omega$ is the unique homomorphism satisfying $k \circ \overline{h} = F\overline{h} \circ \lambda^T_\Omega \circ Tk$. The statement thus follows from uniqueness:

$$\begin{array}{ccccc}
T\Omega & \xrightarrow{\alpha_\Omega} & S\Omega & \xrightarrow{h} & \Omega \\
{\scriptstyle Tk}\downarrow & & \downarrow{\scriptstyle Sk} & & \\
TF\Omega & \xrightarrow{\alpha_{F\Omega}} & SF\Omega & & \downarrow{\scriptstyle k} \\
{\scriptstyle \lambda^T_\Omega}\downarrow & & \downarrow{\scriptstyle \lambda^S_\Omega} & & \\
FT\Omega & \xrightarrow[F\alpha_\Omega]{} & FS\Omega & \xrightarrow[Fh]{} & F\Omega
\end{array} .$$

□

Lemma 4.6.0.7. *Let $\mathcal{X}$ be a T-succinct automaton, then $\mathsf{obs}^\dagger_{\mathcal{X}} = \mathsf{obs}^\dagger_{\mathsf{ext}(\mathcal{X})}$.*

Proof. Since by Proposition 4.6.0.1 the morphism $\alpha_X : \mathsf{exp}_T(\mathcal{X}) \to \alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))$ is a λ^T-bialgebra homomorphism, we have by uniqueness $\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))} \circ \alpha_X = \mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}$. Since any final bialgebra homomorphism is induced by the underlying final F-coalgebra homomorphism it holds $\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))} = \mathsf{obs}_{\mathsf{exp}(\mathsf{ext}_S(\mathcal{X}))}$, which thus implies

$$\mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{X}))} \circ \alpha_X = \mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}. \tag{4.3}$$

The statement follows from

$$\begin{aligned}
\mathsf{obs}^\dagger_{\mathcal{X}} &= \mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})} \circ \eta^T_X && \text{(Definition of } \mathsf{obs}^\dagger_{\mathcal{X}}) \\
&= \mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{X}))} \circ \alpha_X \circ \eta^T_X && (4.3) \\
&= \mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{X}))} \circ \eta^S_X && (\alpha \text{ is distributive law hom.})
\end{aligned}$$

$$= \mathsf{obs}^{\dagger}_{\mathsf{ext}(\mathcal{X})} \qquad \text{(Definition of } \mathsf{obs}^{\dagger}_{\mathsf{ext}(\mathcal{X})}\text{)}$$

□

Lemma 4.6.0.8. *Let $\alpha : \lambda^S \to \lambda^T$ be a distributive law homomorphism. For any T-succinct automaton $\mathcal{X}$ it holds that $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) = \mathsf{im}(h \circ \alpha_\Omega \circ T(\mathsf{obs}^{\dagger}_{\mathcal{X}}))$ and $\mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))}) = \mathsf{im}(h \circ S(\mathsf{obs}^{\dagger}_{\mathcal{X}}))$, where (Ω, h, k) is the final λ^S-bialgebra.*

Proof. By Lemma 4.6.0.6 $(\Omega, h \circ \alpha_\Omega, k)$ is the final λ^T-bialgebra. The first statement follows from

$$\begin{aligned}
\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})} &= \mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})} \circ \mu^T_X \circ T(\eta^T_X) && \text{(Definition of } T\text{)} \\
&= h \circ \alpha_\Omega \circ T(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \circ T(\eta^T_X) && (\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})} \text{ is } T\text{-algebra hom}) \\
&= h \circ \alpha_\Omega \circ T(\mathsf{obs}^{\dagger}_{\mathcal{X}}) && \text{(Definition of } \mathsf{obs}^{\dagger}_{\mathcal{X}}\text{)}
\end{aligned}$$

Similarly one shows that $\mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{X}))} = h \circ S(\mathsf{obs}^{\dagger}_{\mathsf{ext}(\mathcal{X})})$. Since any final bialgebra homomorphism is induced by the underlying final F-coalgebra homomorphism, it thus follows

$$\begin{aligned}
\mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))}) &= \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{X}))}) && (\mathsf{obs}_{\alpha(\mathsf{exp}_S(\mathsf{ext}(\mathcal{X})))} = \mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{X}))}) \\
&= \mathsf{im}(h \circ S(\mathsf{obs}^{\dagger}_{\mathsf{ext}(\mathcal{X})})) && (\mathsf{obs}_{\mathsf{exp}_S(\mathsf{ext}(\mathcal{X}))} = h \circ S(\mathsf{obs}^{\dagger}_{\mathsf{ext}(\mathcal{X})})) \\
&= \mathsf{im}(h \circ S(\mathsf{obs}^{\dagger}_{\mathcal{X}})) && \text{(Lemma 4.6.0.7)}
\end{aligned}$$

□

In light of Lemma 4.6.0.3, the above Lemma 4.6.0.8 implies that a T-succinct automaton $\mathcal{X}$ is α-closed iff there exists an isomorphism $\mathsf{im}(h \circ \alpha_\Omega \circ T(\mathsf{obs}^{\dagger}_{\mathcal{X}})) \cong \mathsf{im}(h \circ S(\mathsf{obs}^{\dagger}_{\mathcal{X}}))$.

4.6.1 Applications to Canonical Automata

In this section we instantiate Theorem 4.6.0.5 to characterise a variety of canonical acceptors from the literature as size-minimal representatives among subclasses of α-closed succinct automata, i.e. those automata whose images of the two semantics

induced by α coincide. We begin with the canonical RFSA and the minimal xor automaton, for which α is the identitity and α-closedness therefore is trivial.

In [49] the canonical RFSA for L has been characterised as size-minimal among those NFAs accepting L for which states accept a residual of L. More recently, it was shown that the class in fact can be extended to those NFAs accepting L for which states accept a *union* of residuals of L [119]. In Corollary 4.6.1.3 we recover the latter as a consequence of the second point in Theorem 4.6.0.5. To prove Corollary 4.6.1.3, we need the following technical result.

Lemma 4.6.1.1. *Let $\mathbb{X} = (X, h, k)$ be an observable λ^S-bialgebra and $\mathbb{G}$ a generator for $(X, h \circ \alpha_X)$, then $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathsf{gen}(\alpha(\mathbb{X}),\mathbb{G}))}) \simeq X$.*

Proof. By Proposition 4.4.0.5 there exists a surjective λ^T-bialgebra homomorphism $\mathsf{exp}_T(\mathsf{gen}(\alpha(\mathbb{X}), \mathbb{G})) \to \alpha(\mathbb{X})$. Since the final λ^T-bialgebra homomorphism is induced by the underlying final F-coalgebra homomorphism and $\alpha(\mathbb{X}) = (X, h \circ \alpha_X, k)$, it holds $\mathsf{obs}_{\alpha(\mathbb{X})} = \mathsf{obs}_{\mathbb{X}}$. The statement follows from the uniqueness of epi-mono factorisations and the definition of $\alpha(\mathbb{X})$:

$$\begin{array}{ccc} \mathsf{exp}_T(\mathsf{gen}(\alpha(\mathbb{X}), \mathbb{G})) & \twoheadrightarrow & \alpha(\mathbb{X}) \\ \downarrow & \swarrow \simeq & \downarrow{\scriptstyle \mathsf{obs}_{\alpha(\mathbb{X})} = \mathsf{obs}_{\mathbb{X}}} \\ \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathsf{gen}(\alpha(\mathbb{X}),\mathbb{G}))}) & \hookrightarrow & \Omega \end{array} .$$

□

We write $\overline{Y}^{\mathbb{A}}$ for the algebraic closure of a subset $Y \subseteq A$ of some T-algebra $\mathbb{A}$ (for details see Section 5.2). For example, if $Y = \mathsf{im}(f)$ for some f with codomain $\mathbb{A} = (A, h)$, the closure is given by the induced T-algebra structure on $\mathsf{im}(h \circ Tf)$. Recall that the set $2 = \{0, 1\}$ – and consequently the set 2^{A^*} underlying the final coalgebra for the functor $FX = 2 \times X^A$ – can be turned into a $\mathcal{P}$-algebra (Lemma 4.3.0.3), an $\mathcal{H}$-algebra (Lemma 4.3.0.4), an $\mathcal{A}$-algebra (Lemma 4.3.0.5), and an $\mathcal{R}$-algebra (Lemma 4.3.0.6). We thus can take the closure of a subset $Y \subseteq 2^{A^*}$ with respect to a T-algebra structure $(2^{A^*}, h_T)$, for any $T \in \{\mathcal{P}, \mathcal{H}, \mathcal{A}, \mathcal{R}\}$. In such a situation, we keep h_T implicit and abbreviate $\overline{Y}^{\mathscr{C}^T} := \overline{Y}^{(2^{A^*}, h_T)}$. For example, $\overline{\mathsf{Der}(L)}^{\mathsf{CSL}}$ denotes the $\mathcal{P}$-closure of the subset $\mathsf{Der}(L) \subseteq 2^{A^*}$. For more cases recall Example 2.2.0.11.

Corollary 4.6.1.2. *Let $\alpha : \mathcal{P}X \to \mathcal{H}X$ satisfy $\alpha_X(\varphi)(\psi) = \vee_{x\in X}\varphi(x) \wedge \psi(x)$. For any unpointed non-deterministic automaton $\mathcal{X}$ it holds:*

- $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{X})}) = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathsf{CSL}}$;
- $\mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_{\mathcal{H}}(\mathsf{ext}(\mathcal{X})))}) = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathsf{CABA}}$.

Proof. The final $\lambda^{\mathcal{H}}$-bialgebra is given by $(2^{A^*}, 2^{\eta^{\mathcal{H}}_{A^*}}, (\varepsilon, \delta))$. In the proof of Lemma 4.5.3.2 it was shown that $2^{\eta^{\mathcal{H}}_X} \circ \alpha_{\mathcal{P}X} = \mu^{\mathcal{P}}_X$. Thus it follows

$$\begin{aligned}
\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{X})}) &= \mathsf{im}(2^{\eta^{\mathcal{H}}_{A^*}} \circ \alpha_{2^{A^*}} \circ \mathcal{P}(\mathsf{obs}^{\dagger}_{\mathcal{X}})) && \text{(Lemma 4.6.0.8)}\\
&= \mathsf{im}(\mu^{\mathcal{P}}_{A^*} \circ \mathcal{P}(\mathsf{obs}^{\dagger}_{\mathcal{X}})) && (2^{\eta^{\mathcal{H}}_X} \circ \alpha_{\mathcal{P}X} = \mu^{\mathcal{P}}_X)\\
&= \{\cup_{u\in U}\mathsf{obs}^{\dagger}_{\mathcal{X}}(u) \mid U \subseteq X\} && \text{(Definitions of } \mathcal{P}(-),\ \mu^{\mathcal{P}})\\
&= \overline{\{\mathsf{obs}^{\dagger}_{\mathcal{X}}(x) \mid x \in X\}}^{\mathsf{CSL}} && \text{(Definition of } \overline{(-)}^{\mathsf{CSL}})
\end{aligned}$$

Similarly one computes

$$\begin{aligned}
&\mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_{\mathcal{H}}(\mathsf{ext}(\mathcal{X})))})\\
&= \text{(Lemma 4.6.0.8)}\\
&\mathsf{im}(2^{\eta^{\mathcal{H}}_{A^*}} \circ \mathcal{H}(\mathsf{obs}^{\dagger}_{\mathcal{X}}))\\
&= \text{(Definitions of } 2^{\eta^{\mathcal{H}}_{A^*}}, \mathcal{H}(-))\\
&\{\cup_{\varphi\in\Phi} \cap_{x\in\varphi} \mathsf{obs}^{\dagger}_{\mathcal{X}}(x) \cap \cap_{x\notin\varphi}\mathsf{obs}^{\dagger}_{\mathcal{X}}(x)^c \mid \Phi \subseteq 2^X\}\\
&= \text{(Set equality)}\\
&\{\{w \in A^* \mid \{x \in X \mid \mathsf{obs}^{\dagger}_{\mathcal{X}}(x)(w) = 1\} \in \Phi\} \mid \Phi \subseteq 2^X\}\\
&= \text{(Definition of } \overline{(-)}^{\mathsf{CABA}})\\
&\overline{\{\mathsf{obs}^{\dagger}_{\mathcal{X}}(x) \mid x \in X\}}^{\mathsf{CABA}}
\end{aligned}$$

□

Corollary 4.6.1.3. *The canonical RFSA for L is size-minimal among NFAs $\mathcal{Y}$ accepting L with $\overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathsf{CSL}} \subseteq \overline{\mathsf{Der}(L)}^{\mathsf{CSL}}$.*

Proof. By Lemma 4.3.0.3 the morphism $h^{\mathcal{P}} : \mathcal{P}2 \to 2$ with $h^{\mathcal{P}}(\varphi) = \varphi(1)$ is a $\mathcal{P}$-algebra. As shown in Lemma 4.3.0.2, it can used to derive a canonical distributive law $\lambda^{\mathcal{P}}$. It is not hard to see that the minimal pointed $\lambda^{\mathcal{P}}$-bialgebra $\mathbb{M}$ accepting L exists and that its underlying state space is given by the finite complete join-semi lattice $\overline{\mathsf{Der}(L)}^{\mathsf{CSL}}$. By Lemma 4.4.0.3 the join-irreducibles for $\mathbb{M}$ constitute a size-minimal generator $\mathbb{G}$. By definition, the canonical RFSA for L is given by $\mathcal{X} := \mathsf{gen}(\mathbb{M}, \mathbb{G})$. From Lemma 4.6.1.1 it follows that $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{X})}) \simeq \overline{\mathsf{Der}(L)}^{\mathsf{CSL}}$. As seen in e.g. Corollary 4.6.1.2, one has $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{Y})}) = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathsf{CSL}}$ for any NFA $\mathcal{Y}$. By choosing α as the identity, which implies α-closedness for any NFA, the statement thus follows from Theorem 4.6.0.5. □

The second condition in Theorem 4.6.0.5 is always satisfied for a *reachable* succinct automaton $\mathcal{Y}$. Since for $\mathbb{Z}_2$-weighted automata it is possible to find an equivalent reachable $\mathbb{Z}_2$-weighted automaton with less or equally many states (which for NFA is not necessarily the case), the minimal xor automaton is minimal among *all* $\mathbb{Z}_2$-weighted automata, as was already known from for instance [156].

Corollary 4.6.1.4. *The minimal xor automaton for L is size-minimal among $\mathbb{Z}_2$-weighted automata accepting L.*

Proof. Analogous to Corollary 4.6.1.3 one can show that the minimal xor automaton for $L \subseteq A^*$ is size-minimal among all $\mathbb{Z}_2$-weighted automata $\mathcal{Y}$ accepting L such that $\overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathbb{Z}_2\text{-}\mathsf{Vect}} \subseteq \overline{\mathsf{Der}(L)}^{\mathbb{Z}_2\text{-}\mathsf{Vect}}$. Specific to this case are Lemma 4.3.0.6 and Lemma 4.4.0.4. It remains to observe that for any $\mathbb{Z}_2$-weighted automaton $\mathcal{X}$, one can find an equivalent $\mathbb{Z}_2$-weighted automaton $\mathcal{Y}$ with a state space of size not greater than the one of $\mathcal{X}$, such that above inclusion holds. The state space of $\mathcal{Y}$ can be chosen as a basis for the underlying vector space of the epi-mono factorisation of the reachability map $\mathcal{X}(A^*) \to \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathbb{Z}_2\text{-}\mathsf{Vect}}$. □

For the átomaton, the distromaton, and the minimal xor-CABA automaton the distributive law homomorphism α in play is non-trivial; α-closedness translates to the below equalities between closures. In all three cases it is possible to waive the inclusion induced by the second point in Theorem 4.6.0.5. The proof of the minimality result

for the átomaton (Corollary 4.6.1.8) requires the following three technical results (Corollary 4.6.1.5, Corollary 4.6.1.2, Corollary 4.6.1.7).

Corollary 4.6.1.5. *Let $\alpha_X : \mathcal{P}X \to \mathcal{H}X$ satisfy $\alpha_X(\varphi)(\psi) = \vee_{x\in X}\varphi(x) \wedge \psi(x)$. If $B = (X, h)$ is a finite $\mathcal{H}$-algebra, then $(\mathsf{At}(B), i, d)$ with $i(a) = a$ and $d(x) = \{a \in \mathsf{At}(B) \mid a \leq x\}$ is a size-minimal generator for $(X, h \circ \alpha_X)$.*

Proof. By Lemma 4.5.3.2 $(\mathsf{At}(B), i, d)$ is a basis for $(X, h \circ \alpha_X)$. Analogously to Lemma 4.4.0.4, it follows that any finite basis for a $\mathcal{P}$-algebra is a size-minimal generator. □

Lemma 4.6.1.6 ([121])**.** *Let A be a sub-lattice of a finite distributive lattice B, then $|J(A)| \leq |J(B)|$.*

Proof. For $x \in J(B)$ define $\hat{x} := \bigwedge\{y \in A \mid x \leq y\} \geq x$. To see that $\hat{x} \in J(A)$, assume $\hat{x} = y \vee z$ for $y, z \in A$. By distributivity we have $x = \hat{x} \wedge x = (y \vee z) \wedge x = (y \wedge x) \vee (z \wedge x)$. Since $x \in J(B)$, it thus follows w.l.o.g. $x = y \wedge x$, which implies $x \leq y$. Consequently $\hat{x} \leq y \leq \hat{x}$, i.e. $\hat{x} = y$. Let $z \in J(A)$, then the join-density of join-irreducibles implies

$$z = \vee\{x \in J(B) \mid x \leq z\} = \vee\{\hat{x} \in J(A) \mid x \in J(B) : x \leq z\}.$$

Since z is join-irreducible it follows $z = \widehat{x_z}$ for some $x_z \in J(B)$ with $x_z \leq z$. We thus find $J(A) = \{\hat{x} \mid x \in J(B)\}$, which implies the claim $|J(A)| \leq |J(B)|$. □

Corollary 4.6.1.7. *Let A be a sub-algebra of a finite atomic Boolean algebra B. Then $|\mathsf{At}(A)| \leq |\mathsf{At}(B)|$.*

Proof. For atomic Boolean algebras, join-irreducibles and atoms coincide. Every Boolean algebra is in particular a distributive lattice. The claim thus follows from Lemma 4.6.1.6. □

Corollary 4.6.1.8. *The átomaton for L is size-minimal among non-deterministic automata $\mathcal{Y}$ accepting L with $\overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CSL}} = \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CABA}}$.*

Proof. Let $\alpha : \lambda^{\mathcal{H}} \to \lambda^{\mathcal{P}}$ be the distributive law homomorphism in Corollary 4.5.3.1.

- By Corollary 4.6.1.2 and the definition of α-closedness, any non-deterministic automaton $\mathcal{Y}$ satisfies $\overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CSL}} = \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CABA}}$ iff it is α-closed.

- Let $\mathbb{M}$ be the minimal pointed $\lambda^{\mathcal{H}}$-bialgebra accepting L. The state-space of $\mathbb{M}$ is the finite set $\overline{\mathsf{Der}(L)}^{\mathsf{CABA}}$. By Corollary 4.6.1.5, the set $\mathsf{At}(\overline{\mathsf{Der}(L)}^{\mathsf{CABA}})$ underlies a size-minimal generator $\mathbb{G}$ for the algebraic part of $\alpha(\mathbb{M})$. The átomaton for L can thus be recovered as the α-closed non-deterministic automaton $\mathcal{X} := \mathsf{gen}(\alpha(\mathbb{M}), \mathbb{G})$ accepting L in Theorem 4.6.0.5. In particular, by the first bullet point above, the átomaton thus lives in the class of non-deterministic automata $\mathcal{Y}$ accepting L with $\overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CSL}} = \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CABA}}$.

- Let $\mathcal{Y}$ be any non-deterministic automaton accepting L with $\overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CSL}} = \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CABA}}$ and state-space Y. By construction, there exists an epimorphism $\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{Y})} : \mathcal{P}Y \twoheadrightarrow \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CSL}}$, which turns Y into a generator for the finite $\mathcal{P}$-algebra $B := \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CSL}} = \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{Y}})}^{\mathsf{CABA}}$. As for CABAs join-irreducibles and atoms coincide, the size-minimality of join-irreducibles in Lemma 4.4.0.3 thus implies $|\mathsf{At}(B)| \leq |Y|$. By Theorem 4.6.0.5, we have that $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \subseteq B$, where $\mathcal{X}$ denotes the átomaton. From Lemma 4.6.1.1 and the definition of $\mathcal{X}$ it follows that $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_T(\mathcal{X})}) \cong \overline{\mathsf{Der}(L)}^{\mathsf{CABA}}$. We thus have $\overline{\mathsf{Der}(L)}^{\mathsf{CABA}} \subseteq B$. By Corollary 4.6.1.7, it follows $|\mathsf{At}(\overline{\mathsf{Der}(L)}^{\mathsf{CABA}})| \leq |\mathsf{At}(B)|$. Consequently we can deduce $|\mathsf{At}(\overline{\mathsf{Der}(L)}^{\mathsf{CABA}})| \leq |Y|$, which shows that the átomaton is size-minimal. □

The above result can be shown to be similar to [119, Theorem 4.9], which characterises the átomaton as size-minimal among non-deterministic automata whose accepted languages are *closed under complement*.

Corollary 4.6.1.10 below is very similar to a characterisation of the distromaton as size-minimal among non-deterministic automata whose accepted languages are *closed under intersection* [119, Theorem 4.13]. The proof of Corollary 4.6.1.10 requires the following technical result.

Corollary 4.6.1.9. *Let $\alpha : \mathcal{P}X \to \mathcal{A}X$ satisfy $\alpha_X(\varphi)(\psi) = \vee_{x \in X} \varphi(x) \wedge \psi(x)$. For any unpointed non-deterministic automaton $\mathcal{X}$ it holds:*

- $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{X})}) = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathsf{CSL}}$;
- $\mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_{\mathcal{A}}(\mathsf{ext}(\mathcal{X})))}) = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathsf{CDL}}$.

Proof. Analogous to the proof of Corollary 4.6.1.2. □

Corollary 4.6.1.10. *The distromaton for L is size-minimal among non-deterministic automata $\mathcal{Y}$ accepting L with* $\overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathsf{CSL}} = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathsf{CDL}}$.

Proof. Analogous to the proof of Corollary 4.6.1.8. Specific to this case are the results Lemma 4.3.0.3, Lemma 4.3.0.5, Corollary 4.5.4.1, Lemma 4.4.0.3, Lemma 4.6.1.6, and Corollary 4.6.1.9. □

The size-minimality result (Corollary 4.6.1.12) for the newly discovered minimal xor-CABA automaton is analogous to the ones for the átomaton and the distromaton. It requires the following technical result.

Corollary 4.6.1.11. *Let $\alpha : \mathcal{R}X \to \mathcal{H}X$ satisfy $\alpha_X(\varphi)(\psi) = \bigoplus_{x \in X} \varphi(x) \cdot \psi(x)$. For any unpointed $\mathbb{Z}_2$-weighted automaton $\mathcal{X}$ it holds:*

- $\mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{R}}(\mathcal{X})}) = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathbb{Z}_2\text{-}\mathsf{Vect}}$;
- $\mathsf{im}(\mathsf{obs}_{\alpha(\mathsf{exp}_{\mathcal{H}}(\mathsf{ext}(\mathcal{X})))}) = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathsf{CABA}}$.

Proof. Analogous to the proof of Corollary 4.6.1.2. □

Corollary 4.6.1.12. *The minimal xor-CABA automaton for L is size-minimal among $\mathbb{Z}_2$-weighted automata $\mathcal{Y}$ accepting L with* $\overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathbb{Z}_2\text{-}\mathsf{Vect}} = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathsf{CABA}}$.

Proof. Analogous to the proof of Corollary 4.6.1.8. Specific to this case are Lemma 4.3.0.4, Lemma 4.3.0.6, Corollary 4.5.5.1, Lemma 4.4.0.4, Corollary 4.6.1.11 and the observation that if $A \subseteq B$ is a subvector space of a finite vector space B, then $\mathsf{dim}(A) \leq \mathsf{dim}(B)$. □

We conclude with a size-comparison between acceptors that is parametric in the closure of derivatives.

Corollary 4.6.1.13.

- *If* $\overline{\mathsf{Der}(L)}^{\mathbb{Z}_2\text{-}\mathsf{Vect}} = \overline{\mathsf{Der}(L)}^{\mathsf{CABA}}$*, then the minimal xor automaton and the minimal xor-CABA automaton for* L *are of the same size.*
- *If* $\overline{\mathsf{Der}(L)}^{\mathsf{CSL}} = \overline{\mathsf{Der}(L)}^{\mathsf{CDL}}$*, then the canonical RFSA and the distromaton for* L *are of the same size.*
- *If* $\overline{\mathsf{Der}(L)}^{\mathsf{CSL}} = \overline{\mathsf{Der}(L)}^{\mathsf{CABA}}$*, then the canonical RFSA and the átomaton for* L *are of the same size.*

Proof.

- By Corollary 4.6.1.4 the minimal xor automaton $\mathcal{X}$ is of size not greater than the minimal xor-CABA automaton $\mathcal{Y}$. Conversely, we find

$$\begin{aligned}
\overline{\mathsf{Der}(L)}^{\mathsf{CABA}} &= \overline{\mathsf{Der}(L)}^{\mathbb{Z}_2\text{-}\mathsf{Vect}} && \text{(Assumption)}\\
&= \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{R}}(\mathcal{X})}) && \text{(Lemma 4.6.1.1)}\\
&= \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathbb{Z}_2\text{-}\mathsf{Vect}} && \text{(Corollary 4.6.1.11)}
\end{aligned}$$

  which can be used to show $\overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathbb{Z}_2\text{-}\mathsf{Vect}} = \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathsf{CABA}}$. By Corollary 4.6.1.12 the latter implies that $\mathcal{Y}$ is of size not greater than $\mathcal{X}$, which shows the claim.

- Let $\mathcal{X}$ denote the canonical RFSA and $\mathcal{Y}$ the distromaton. On the one hand we find

$$\begin{aligned}
\overline{\mathsf{Der}(L)}^{\mathsf{CSL}} &= \overline{\mathsf{Der}(L)}^{\mathsf{CDL}} && \text{(Assumption)}\\
&= \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{Y})}) && \text{(Lemma 4.6.1.1)}\\
&= \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{Y}})}^{\mathsf{CSL}} && \text{(Corollary 4.6.1.9)}
\end{aligned}$$

  which by Corollary 4.6.1.3 implies that $\mathcal{X}$ is of size not greater than $\mathcal{Y}$. Conversely, we establish the equality

$$\begin{aligned}
\overline{\mathsf{Der}(L)}^{\mathsf{CDL}} &= \overline{\mathsf{Der}(L)}^{\mathsf{CSL}} && \text{(Assumption)}\\
&= \mathsf{im}(\mathsf{obs}_{\mathsf{exp}_{\mathcal{P}}(\mathcal{X})}) && \text{(Lemma 4.6.1.1)}\\
&= \overline{\mathsf{im}(\mathsf{obs}^{\dagger}_{\mathcal{X}})}^{\mathsf{CSL}} && \text{(Corollary 4.6.1.9)}
\end{aligned}$$

which can be used to show $\overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{X}})}^{\mathsf{CSL}} = \overline{\mathsf{im}(\mathsf{obs}^\dagger_{\mathcal{X}})}^{\mathsf{CDL}}$. By Corollary 4.6.1.10 the latter implies that $\mathcal{Y}$ is of size not greater than $\mathcal{X}$, which shows the claim.

- The proof for the átomaton is analogous to the proof for the distromaton in the previous point. □

4.7 Related Work

One of the motivations for our work are active learning algorithms for the derivation of succinct state-based models [14]. A major challenge in learning non-deterministic models is the lack of a canonical target acceptor for a given language [49]. The problem has been independently approached for different variants of non-determinism, often with the idea of finding a subclass admitting a unique representative [53, 28] such as the canonical RFSA, the minimal xor automaton, or the átomaton.

A more general and unifying perspective on learning automata that may not have a canonical target was given by Van Heerdt [66, 63, 64]. One of the central notions in this work is the concept of a scoop, originally introduced by Arbib and Manes [19] and here referred to as a generator. The main contribution in [66] is a general procedure to find irreducible sets of generators, which thus restricts the work to the category of sets. In this chapter we generally work over arbitrary categories, although we assume the existence of a minimal set-based generator in Theorem 4.6.0.5. Furthermore, the work of Van Heerdt has no size-minimality results.

Closely related to the content of this chapter is the work of Myers et al. [119], who present a coalgebraic construction for canonical non-deterministic automata. They cover the canonical RFSA, the minimal xor automaton, the átomaton, and the distromaton. The underlying idea in [119] for finding succinct representations is similar to ours: first they build the minimal DFA for a regular language in a locally finite variety, then they apply an equivalence between the category of finite algebras and a suitable category of finite structured sets and relations. On the one hand, the category of finite algebras in a locally finite variety can be translated into our setting by considering a category of algebras over a monad preserving finite sets. In fact, modulo this translation, many of the categories considered here already appear in [119], e.g.

vector spaces, Boolean algebras, complete join-semi lattices, and distributive lattices. On the other hand, their construction seems to be restricted to the category of sets and non-deterministic automata, while we work over arbitrary monads on arbitrary categories. Their work does not provide a general algorithm to construct a succinct automaton, i.e., the specifics vary with the equivalences considered, while we give a general definition and a soundness argument in Corollary 4.5.1.3. While Myers et al. [119] give minimality results for a wide range of acceptors, each proof follows case-specific arguments. In Theorem 4.6.0.5 we provide a unifying minimality result for succinct automata that implies Corollary 4.6.1.8 (cf. [119, Theorem 4.9]), Corollary 4.6.1.4 (cf. [119, Theorem 4.10]), Corollary 4.6.1.3 (cf. [119, Corollary 4.11]), Corollary 4.6.1.10 (cf. [119, Theorem 4.13]).

4.8 Discussion and Future Work

We have presented a general categorical framework based on bialgebras and distributive law homomorphisms for the derivation of canonical automata. The framework instantiates to a wide range of well-known examples from the literature and allowed us to discover a previously unknown canonical acceptor for regular languages. Finally, we presented a theorem that subsumes previously independently proven minimality results for canonical acceptors, implied new characterisations, and allowed us to make size-comparisons between canonical automata.

In the future, we would like to cover other examples, such as the canonical probabilistic RFSA [53] and the canonical alternating RFSA [28, 17]. Probabilistic automata of the type in [53] are typically modelled as TF-coalgebras instead of FT-coalgebras [81], and thus will need a shift in perspective. For alternating RFSAs we expect a canonical form can be constructed in the spirit of this work, from generators for algebras over the neighbourhood monad, by interpreting the join-dense atoms of a CABA as a full meet of ground elements.

Generally, it would be valuable to have a more systematic treatment of the range of available monads and distributive law homomorphisms [162], making use of the fact that distributive law homomorphisms compose.

Further generalisation in another direction could be achieved by distributive laws

between monads and endofunctors on different categories. For instance, we expect that operations on automata as the product can be captured by homomorphisms between distributive laws of such more general type.

Finally, we would like to lift existing double-reversal characterisations of the minimal DFA [38], the átomaton [39], the distromaton [119], and the minimal xor automaton [156] to general canonical automata. The work in [34, 33] gives a coalgebraic generalisation of Brzozowski's algorithm based on dualities between categories, but does not cover the cases we are interested in. The framework in [6] recovers the átomaton as the result of a minimisation procedure, but does not consider other canonical acceptors.

Chapter 5

Generating Monadic Closures

We have seen that every regular language admits a unique size-minimal deterministic acceptor, while there can be several size-minimal non-deterministic acceptors that are not isomorphic (cf. Figure 4.1). We also have seen that to tackle this issue, authors have identified a number of sub-classes of non-deterministic automata, all admitting canonical minimal representatives. In Chapter 4 we demonstrated that such representatives can generally be recovered categorically in two steps. First, one constructs the minimal bialgebra accepting a given regular language, by closing the minimal coalgebra with additional algebraic structure over a monad. Second, one identifies canonical generators for the algebraic part of the bialgebra, to derive an equivalent coalgebra with side effects in a monad. In this chapter, we further develop the general theory underlying these two steps. On the one hand, we show that deriving a minimal bialgebra from a minimal coalgebra can be realized by applying a monad on an appropriate category of subobjects. On the other hand, we explore the abstract theory of generators and bases for algebras over a monad.

5.1 Introduction

Recall that the framework for the construction of canonical automata in Chapter 4 adopts the well-known representation of automata as coalgebras (Definition 2.2.0.12) and side-effects like non-determinism as monads (Definition 2.2.0.7). For instance, an NFA (without initial states) is represented as a coalgebra $k\colon X \to 2 \times \mathcal{P}(X)^A$ with

side-effects in the powerset monad $\mathcal{P}$.

To derive canonical non-deterministic acceptors, we suggested an idea that is closely related to the (generalised) *powerset construction* of Example 4.3.0.7. Under the latter construction, a coalgebra $k : X \to FTX$ with dynamics in a functor F and side-effects in a monad T is transformed into an equivalent coalgebra $k^\sharp : TX \to FTX$ [140]. The deterministic automata resulting from such determinisation constructions have *additional algebraic structure*: the state-space TX defines a (free) algebra for the monad T, and $k^\sharp$ is a T-algebra homomorphism, thus constituting a *bialgebra* over a distributive law relating F and T (Definition 4.3.0.8). Using the powerset construction, we were able to obtain a canonical succinct acceptor for a regular language $L \subseteq A^*$ by consecutively following two steps.

First, we constructed the minimal[1] pointed coalgebra M_L for the functor $F = 2\times(-)^A$ accepting L. We then equipped the former with additional algebraic structure in a monad T by applying the determinisation procedure to M_L when seen as coalgebra with trivial side-effects in T (cf. Corollary 4.3.0.11). By identifying semantically equivalent states, we then derived the minimal[2] (pointed) bialgebra for L.

Second, we exploited the additional algebraic structure underlying the minimal bialgebra for L to "reverse" the generalised determinisation procedure. That is, we identified canonical generators (Definition 4.4.0.1) to derive an equivalent succinct automaton with side-effects in T (cf. Proposition 4.4.0.5).

In this chapter, we further develop the general theory underlying these two steps by making the following contributions, respectively:

- We generalise the closure of a subset of an algebraic structure with respect to the latter as a functor between categories of subobjects relative to a factorisation system (Proposition 5.2.3.1). We then equip the functor with the structure of a monad (Theorem 5.2.4.2). We investigate the closure of a particular subclass of subobjects: the ones that arise by taking the image of a morphism (Lemma 5.2.5.1). We show that deriving a minimal bialgebra from

[1]Minimal in the sense that every state is reachable by an element of A^* and no two different states observe the same language.

[2]Minimal in the sense that every state is reachable by an element of $T(A^*)$ and no two different states observe the same language.

a minimal coalgebra can be realized by applying the monad to a subobject in this particular class (Example 5.2.5.2).

- We define a category of algebras with generators (Definition 5.3.1.1), which is in adjunction with the category of Eilenberg-Moore algebras (Lemma 5.3.1.2), and, under certain assumptions, monoidal (Corollary 5.3.2.2). We generalise the matrix representation theory of vector spaces (Section 5.3.3) and discuss bases for bialgebras, which are algebras over a particular monad (Section 5.3.4). We compare our ideas with an approach that generalises bases as coalgebras (Section 5.3.5). We find that a basis in our sense induces a basis the sense of [77] (Lemma 5.3.5.1), and identify assumptions under which the reverse is true, too. We characterise generators for finitary varieties in the sense of universal algebra (Lemma 5.3.6.1) and relate our work to the theory of locally finitely presentable categories (Proposition 5.3.7.1).

5.2 Step 1: Closure

In this section, we further explore the categorical construction of minimal canonical acceptors given in Chapter 4. In particular, we show that deriving a minimal bialgebra from a minimal coalgebra by closing the latter with additional algebraic structure has a direct analogue in universal algebra: taking the closure of a subset of an algebra.

5.2.1 Factorisation Systems and Subobjects

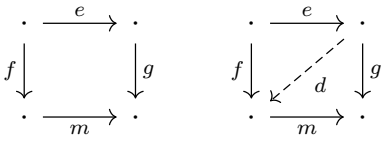
In the category of sets and functions, every morphism can be factored into a surjection onto its image followed by an injection into the codomain of the morphism. In this section, we recall an abstraction of this phenomenon for arbitrary categories. The ideas are well established [36, 129, 106]. We choose to adapt the formalism of [4].

Definition 5.2.1.1 (Factorisation System)**.** Let $\mathscr{E}$ and $\mathscr{M}$ be classes of morphisms in a category $\mathscr{C}$. We call the pair $(\mathscr{E}, \mathscr{M})$ a *factorisation system* for $\mathscr{C}$ if the following three conditions hold:

(F1) Each of $\mathscr{E}$ and $\mathscr{M}$ is closed under composition with isomorphisms.

(F2) Each morphism f in $\mathscr{C}$ can be factored as $f = m \circ e$, with $e \in \mathscr{E}$ and $m \in \mathscr{M}$.

(F3) For each commutative square with $e \in \mathscr{E}$ and $m \in \mathscr{M}$ as on the left below



there exists a unique diagonal d such that the diagram on the right above commutes.

We will use double headed ($\twoheadrightarrow$) and hooked ($\hookrightarrow$) arrows to indicate that a morphism is in $\mathscr{E}$ and $\mathscr{M}$, respectively. If f factors into e and m, we call the codomain of e, or equivalently, the domain of m, the *image* of f and denote it by $\mathsf{im}(f)$.

One can show that each of $\mathscr{E}$ and $\mathscr{M}$ contains all isomorphisms and is closed under composition [4, Prop. 14.6]. From the uniqueness condition on the diagonal one can deduce that factorisations are unique up to unique isomorphism [4, Prop. 14.4]. It further follows that $\mathscr{E}$ has the *right cancellation property*, that is $g \circ f \in \mathscr{E}$ and $f \in \mathscr{E}$ implies $g \in \mathscr{E}$. Dually, $\mathscr{M}$ has the *left cancellation property*, that is, $g \circ f \in \mathscr{M}$ and $g \in \mathscr{M}$ implies $f \in \mathscr{M}$ [4, Prop. 14.9].

As intended, in the category of sets and functions, surjective and injective functions, or equivalently, epi- and monomorphisms, constitute a factorisation system [4, Ex. 14.2]. More involved examples can be constructed for e.g. the category of topological spaces or the category of categories [4, Ex. 14.2]. We are particularly interested in factorisation systems for the category of algebras over a monad.

The naive categorical generalisation of a subset $Y \subseteq X$ is a monomorphism $Y \to X$. Since in the category of sets epi- and monomorphism constitute a factorisation system, we may generalise subsets to arbitrary categories $\mathscr{C}$ with a factorisation system $(\mathscr{E}, \mathscr{M})$ in the following way:

Definition 5.2.1.2 (Subobjects)**.** A *subobject* of an object $X \in \mathscr{C}$ is a morphism $m_Y : Y \hookrightarrow X \in \mathscr{M}$. A morphism $f : m_{Y_1} \to m_{Y_2}$ between subobjects of X consists of a morphism $f : Y_1 \to Y_2$ such that $m_{Y_2} \circ f = m_{Y_1}$.

The category of (isomorphism classes of) subobjects of X is denoted by $\mathsf{Sub}(X)$.

As $\mathscr{M}$ has the left cancellation property, every morphism between subobjects in fact lies in $\mathscr{M}$. We work with isomorphism classes of subobjects since factorisations of morphisms are only defined up to unique isomorphism. For epi-mono factorisations, there is at most one morphism between any two subobjects, that is, $\mathsf{Sub}(X)$ is simply a partially ordered set.

5.2.2 Factorising Algebra Homomorphisms

In this section, we recall that if one is given a category $\mathscr{C}$ with a factorisation system $(\mathscr{E}, \mathscr{M})$ and a monad T on $\mathscr{C}$ that preserves $\mathscr{E}$ (i.e. satisfies $T(e) \in \mathscr{E}$ for all $e \in \mathscr{E}$), it is possible to lift the factorisation system of the base category $\mathscr{C}$ to a factorisation system on the category of Eilenberg-Moore algebras $\mathscr{C}^T$. The result appears in e.g. [158] and may be extended to algebras over an endofunctor. Alternatively, it can be stated in its dual version: if an endofunctor on $\mathscr{C}$ preserves $\mathscr{M}$, it is possible to lift the factorisation system of $\mathscr{C}$ to the category of coalgebras [95, 158].

The factorisation system we propose for $\mathscr{C}^T$ consists of those algebra homomorphisms, whose underlying morphism lies in $\mathscr{E}$ or $\mathscr{M}$, respectively. Clearly such a system preserves (F1). The next result shows that it also satisfies (F3). Note that the statement is slightly more general, as it holds not just for Eilenberg-Moore algebras over the monad T, but for algebras over the underlying endofunctor.

Lemma 5.2.2.1 ([158, Lem. 3.6])**.** *For each commutative square of homomorphisms between algebras for the endofunctor T as on the left below*

$$\begin{array}{ccc} (A,h_A) & \xrightarrow{e} & (B,h_B) \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ (C,h_C) & \xrightarrow[m]{} & (D,h_D) \end{array} \qquad \begin{array}{ccc} (A,h_A) & \xrightarrow{e} & (B,h_B) \\ {\scriptstyle f}\downarrow & \swarrow{\scriptstyle d} & \downarrow{\scriptstyle g} \\ (C,h_C) & \xrightarrow[m]{} & (D,h_D) \end{array}$$

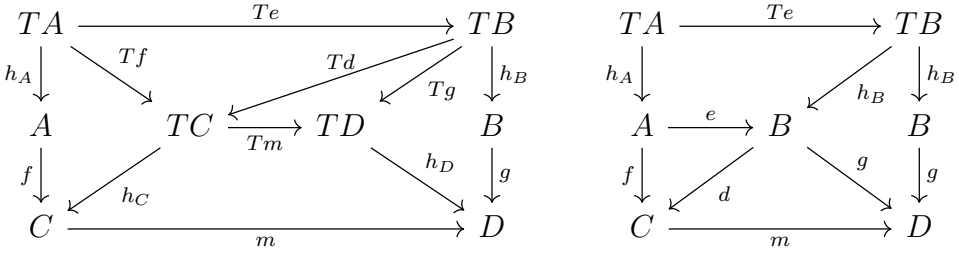
there exists a unique diagonal $d : (B, h_B) \to (C, h_C)$ such that the diagram on the right above commutes.

Proof. The proof for [158, Lem. 3.6] consists of a corollary to the dual statement for coalgebras [158, Lem. 3.3]. Below we offer an explicit version for algebras.

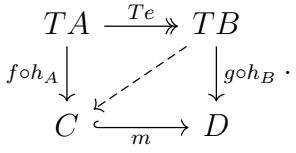
Since any commuting diagram of algebra homomorphism projects to a commuting diagram in $\mathscr{C}$, the factorisation system of $\mathscr{C}$ implies the existence of a unique diagonal d in $\mathscr{C}$. It remains to show that d is an algebra homomorphism, that is, we need to establish the following identity:

$$h_C \circ Td = d \circ h_B.$$

To this end, we observe that since the following two diagrams commute



both $h_C \circ Td$ and $d \circ h_B$ are solutions to the unique diagonal below:



□

Let us now show that the proposed factorisation system satisfies (F2). Assume we are given a homomorphism f as on the left of Figure 5.1. Using the factorisation system of the base category $\mathscr{C}$, we can factorise it, as ordinary morphism, into $e \in \mathscr{E}$ and $m \in \mathscr{M}$. In consequence the outer square of the diagram on the right of Figure 5.1 commutes. Since by assumption the morphism Te is again in $\mathscr{E}$, we thus find a unique diagonal $h_{\mathsf{im}(f)}$ in $\mathscr{C}$ that makes the triangles on the right of Figure 5.1 commute. The result below shows that $h_{\mathsf{im}(f)}$ equips $\mathsf{im}(f)$ with the structure of a T-algebra.

Lemma 5.2.2.2 ([158, Prop. 3.7])**.** $(\mathsf{im}(f), h_{\mathsf{im}(f)})$ *is an Eilenberg-Moore T-algebra.*

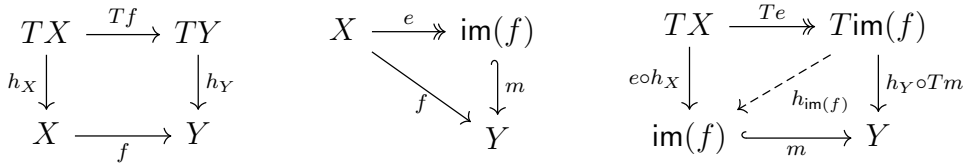


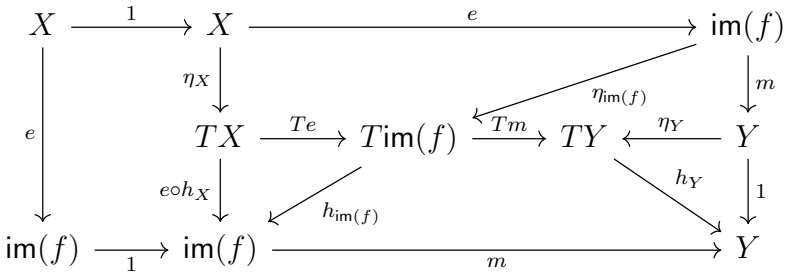
Figure 5.1: Factorising a T-algebra homomorphism via the factorisation system of a base category

Proof. We need to establish the following two identities

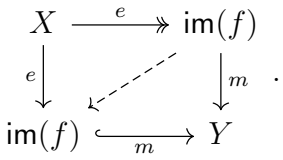
$$h_{\mathsf{im}(f)} \circ \eta_{\mathsf{im}(f)} = \mathsf{id}_{\mathsf{im}(f)}$$
$$h_{\mathsf{im}(f)} \circ \mu_{\mathsf{im}(f)} = h_{\mathsf{im}(f)} \circ T h_{\mathsf{im}(f)}$$

where the latter captures that $h_{\mathsf{im}(f)} : (T\mathsf{im}(f), \mu_{\mathsf{im}(f)}) \to (\mathsf{im}(f), h_{\mathsf{im}(f)})$ is a T-algebra homomorphism.

For the first equality, we observe (as in the proof for ([158, Prop. 3.7])) that since the diagram below commutes

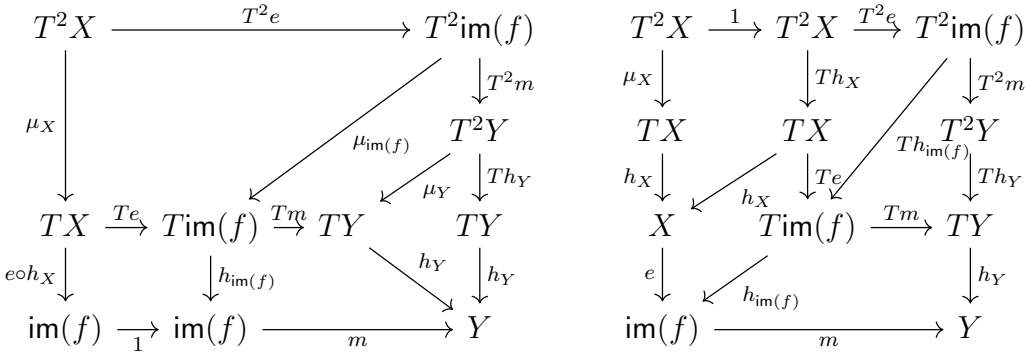


both $h_{\mathsf{im}(f)} \circ \eta_{\mathsf{im}(f)}$ and $\mathsf{id}_{\mathsf{im}(f)}$ are solutions to the unique diagonal in $\mathscr{C}$ below:

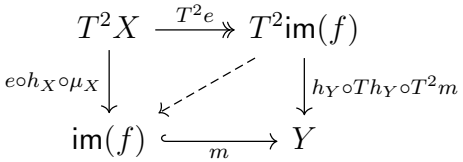


Similarly, for the second equality, we observe that since the following two diagrams

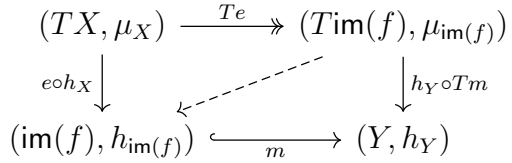
commute



both $h_{_{\mathsf{im}(f)}} \circ \mu_{\mathsf{im}(f)}$ and $h_{\mathsf{im}(f)} \circ Th_{\mathsf{im}(f)}$ are solutions to the unique diagonal below:



Alternatively (as in the proof for [158, Prop. 3.7]), one may observe that since the following outer square of homomorphisms between algebras for the endofunctor T commutes



Lemma 5.2.2.1 implies the existence of a unique diagonal algebra homomorphism making the two triangles above commute. As we know that the diagonal coincides with the unique diagonal of the corresponding diagram in $\mathscr{C}$, which is given by $h_{\mathsf{im}(f)}$, we can deduce that the latter is a T-algebra homomorphism. □

We thus obtain the following factorisation of f into Eilenberg-Moore T-algebra homomorphisms:

$$f = (X, h_X) \overset{e}{\twoheadrightarrow} (\mathsf{im}(f), h_{\mathsf{im}(f)}) \overset{m}{\hookrightarrow} (Y, h_Y).$$

5.2.3 The Subobject Closure Functor

While subobjects in the category of sets generalise subsets, subobjects in the category of algebras generalise subalgebras. By taking the algebraic closure of a subset of an algebra one can thus transition from one category of subobjects to the other.

In this section, we generalise this phenomenon from the category of sets to more general categories. As before, we assume a base category $\mathscr{C}$ with a factorisation system $(\mathscr{E}, \mathscr{M})$ and a monad T on $\mathscr{C}$ that preserves $\mathscr{E}$. Our aim is to construct, for any T-algebra $\mathbb{X}$ with carrier X, a functor from the subobjects $\mathsf{Sub}(X)$ in $\mathscr{C}$ to the subobjects $\mathsf{Sub}(\mathbb{X})$ in $\mathscr{C}^T$ that assigns to a subobject of X its *closure*, that is, the least T-subalgebra of $\mathbb{X}$ containing it.

Recall the natural isomorphism that witnesses the free Eilenberg-Moore algebra adjunction. For any object Y in $\mathscr{C}$ and T-algebra $\mathbb{X} = (X, h)$, it maps a morphism $\varphi : Y \to X$ to the T-algebra homomorphism $\varphi^\sharp := h \circ T\varphi : (TY, \mu_Y) \to (X, h)$. In Section 5.2.2 we have seen that the factorisation system of $\mathscr{C}$ naturally lifts to a factorisation system on the category of T-algebras. In particular, we know that up to isomorphism the homomorphism $\varphi^\sharp$ admits a factorisation of the following form:

$$\varphi^\sharp = (TY, \mu_Y) \overset{e_{\mathsf{im}(\varphi^\sharp)}}{\twoheadrightarrow} (\mathsf{im}(\varphi^\sharp), h_{\mathsf{im}(\varphi^\sharp)}) \overset{m_{\mathsf{im}(\varphi^\sharp)}}{\hookrightarrow} (X, h).$$

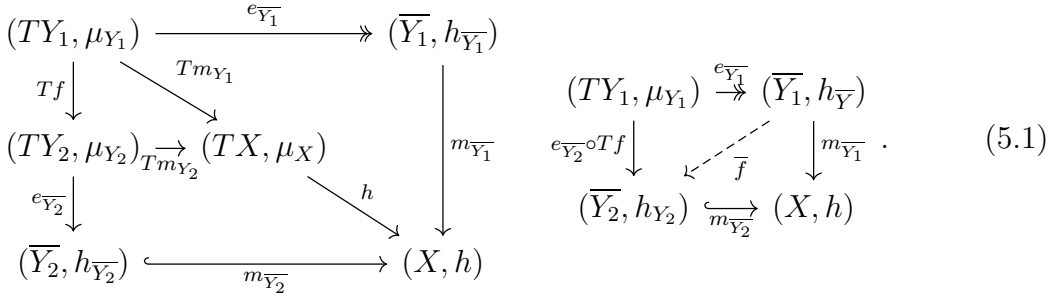
In the case that the morphism φ is given by a subobject

$$m_Y : Y \to X \in \mathscr{M},$$

the above construction yields a second subobject

$$m_{\overline{Y}} : \overline{Y} \to \mathbb{X} \in \mathscr{M},$$

where $\overline{Y} := (\mathsf{im}(m_Y^\sharp), h_{\mathsf{im}(m_Y^\sharp)})$. Since for a morphism $f : m_{Y_1} \to m_{Y_2}$ between subobjects of X the diagram on the left of (5.1) below commutes, there further exists a unique diagonal algebra homomorphism $\overline{f} : m_{\overline{Y_1}} \to m_{\overline{Y_2}}$ between subobjects of $\mathbb{X}$ making the two triangles on the right of (5.1) commute:



(5.1)

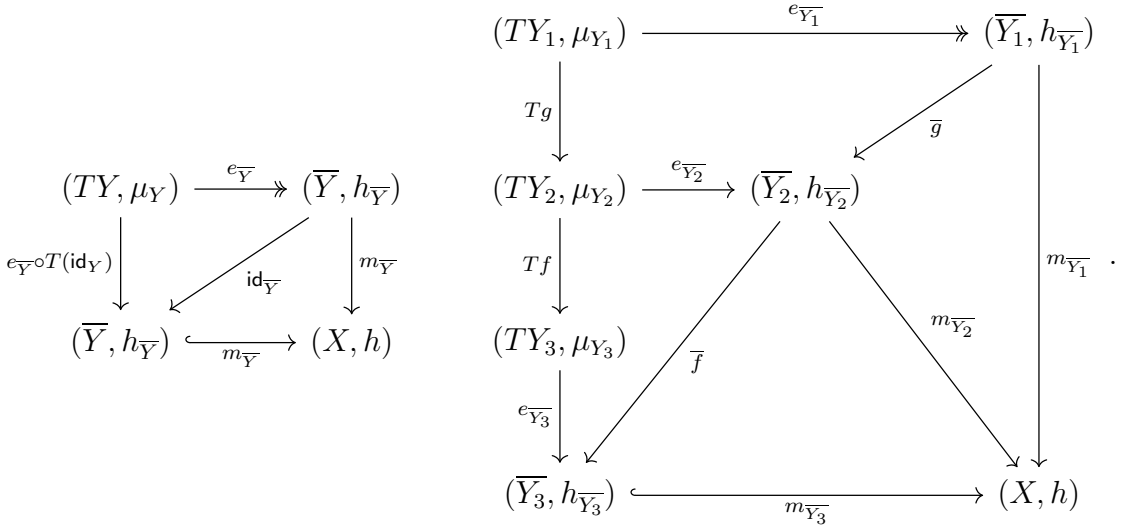
The following result shows that the above constructions are compositional.

Proposition 5.2.3.1. *The assignments $m_Y \mapsto m_{\overline{Y}}$ and $f \mapsto \overline{f}$ yield a functor $\overline{(\cdot)}^{\mathbb{X}} : \mathsf{Sub}(X) \to \mathsf{Sub}(\mathbb{X})$.*

Proof. We need to establish the identities

$$\overline{\mathsf{id}_Y} = \mathsf{id}_{\overline{Y}} \quad \text{and} \quad \overline{f \circ g} = \overline{f} \circ \overline{g}.$$

As before, the equations follow from the uniqueness of diagonals and the commutativity of the left, respectively right, diagram below:



□

Let us instantiate above result for the free vector space monad on the category of sets equipped with its canonical surjective-injective factorisation system.

Example 5.2.3.2. Given an injective embedding m_Y of some set Y into a vector space $\mathbb{V}$, one easily verifies that the lifting $(m_Y)^\sharp$ maps a formal linear combination $\sum_i \lambda_i \cdot y_i$ to the vector $\sum_i \lambda_i \cdot m_Y(y_i)$ in V. The image $\overline{Y}$ is thus given by the vector space that consists of equivalence classes of formal linear combinations, and the injection $m_{\overline{Y}}$ interprets representatives as demonstrated. In particular, if Y is a subset of $\mathbb{V}$, that is, $m_Y(y) = y$ for all $y \in Y$, the closure can be recognised as the sub vector space of $\mathbb{V}$ generated by Y.

Mapping an algebra homomorphism with codomain $\mathbb{X}$ to the $\mathscr{M}$-part of its factorisation extends to a functor from the slice category[3] over $\mathbb{X}$ to the category of subobjects of $\mathbb{X}$. Similarly, one observes that the free Eilenberg-Moore algebra adjunction gives rise to a functor from the slice category over X to the slice category over $\mathbb{X}$. Finally, it is clear that there exists a functor from the category of subobjects of X to the slice category over X. The functor defined in Proposition 5.2.3.1 can thus be recognised as the following composition:

$$\begin{array}{ccc} \mathscr{C}/X & \longrightarrow & \mathscr{C}^T/\mathbb{X} \\ \uparrow & & \downarrow \\ \mathsf{Sub}(X) & \xrightarrow{\overline{(\cdot)}^{\mathbb{X}}} & \mathsf{Sub}(\mathbb{X}) \end{array} . \tag{5.2}$$

5.2.4 The Subobject Closure Monad

In this section, we show that the functor $\overline{(\cdot)}^{\mathbb{X}}$ in Proposition 5.2.3.1 induces a monad on the category of subobjects $\mathsf{Sub}(X)$. As before, we assume a base category $\mathscr{C}$ with a factorisation system $(\mathscr{E}, \mathscr{M})$ and a monad $T = (T, \eta, \mu)$ on $\mathscr{C}$ that preserves $\mathscr{E}$.

We begin by establishing the following two technical identities, which assume a T-algebra $\mathbb{X} = (X, h)$ and a subobject $m_Y : Y \hookrightarrow X \in \mathscr{M}$.

Lemma 5.2.4.1. *The following two equalities hold:*

[3]The *slice category* $\mathscr{C}/X$ of a category $\mathscr{C}$ over an object $X \in \mathscr{C}$ has as objects (isomorphism classes of) morphisms $g_Y : Y \to X$ with codomain X, and a morphism $f : g_{Y_1} \to g_{Y_2}$ consists of a morphism $f : Y_1 \to Y_2$ in $\mathscr{C}$ that satisfies $g_{Y_1} = g_{Y_2} \circ f$.

- $m_{\overline{Y}} \circ e_{\overline{Y}} \circ \eta_Y = m_Y$
- $m_{\overline{Y}} \circ e_{\overline{Y}} \circ \mu_Y = m_{\overline{\overline{Y}}} \circ e_{\overline{\overline{Y}}} \circ Te_{\overline{Y}}$

Proof. For the first identity we observe:

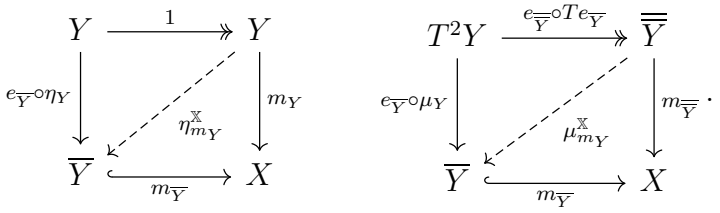
$$\begin{aligned}
m_{\overline{Y}} \circ e_{\overline{Y}} \circ \eta_Y &= m_Y^\sharp \circ \eta_Y && (m_{\overline{Y}} \circ e_{\overline{Y}} = m_Y^\sharp) \\
&= h \circ Tm_Y \circ \eta_Y && (\text{Definition of } m_Y^\sharp) \\
&= h \circ \eta_X \circ m_Y && (\text{Naturality of } \eta) \\
&= m_Y && (h \circ \eta_X = \mathsf{id}_X).
\end{aligned}$$

Similarly, for the second identity we deduce:

$$\begin{aligned}
m_{\overline{Y}} \circ e_{\overline{Y}} \circ \mu_Y &= m_Y^\sharp \circ \mu_Y && (m_{\overline{Y}} \circ e_{\overline{Y}} = m_Y^\sharp) \\
&= h \circ Tm_Y \circ \mu_Y && (\text{Definition of } m_Y^\sharp) \\
&= h \circ \mu_X \circ T^2 m_Y && (\text{Naturality of } \mu) \\
&= h \circ Th \circ T^2 m_Y && (h \circ \mu_X = h \circ Th) \\
&= h \circ Tm_Y^\sharp && (\text{Definition of } m_Y^\sharp) \\
&= h \circ Tm_{\overline{Y}} \circ Te_{\overline{Y}} && (m_Y^\sharp = m_{\overline{Y}} \circ e_{\overline{Y}}) \\
&= m_{\overline{Y}}^\sharp \circ Te_{\overline{Y}} && (\text{Definition of } m_{\overline{Y}}^\sharp) \\
&= m_{\overline{\overline{Y}}} \circ e_{\overline{\overline{Y}}} \circ Te_{\overline{Y}} && (m_{\overline{Y}}^\sharp = m_{\overline{\overline{Y}}} \circ e_{\overline{\overline{Y}}}).
\end{aligned}$$

□

In consequence, we can define candidates for the monad unit $\eta^{\mathbb{X}}$ and the monad multiplication $\mu^{\mathbb{X}}$, respectively, as the unique diagonals below:



(5.3)

By construction both morphisms are homomorphisms of subobjects:

$$\eta^{\mathbb{X}}_{m_Y} : m_Y \longrightarrow m_{\overline{Y}} \qquad \mu^{\mathbb{X}}_{m_Y} : m_{\overline{\overline{Y}}} \longrightarrow m_{\overline{Y}}.$$

The remaining proofs of naturality and the monad laws are covered below. By a slight abuse of notation, we write $\overline{(\cdot)}^{\mathbb{X}}$ for the endofunctor on $\mathsf{Sub}(X)$ that arises by post-composition of the functor in Proposition 5.2.3.1 with the canonical forgetful functor from $\mathsf{Sub}(\mathbb{X})$ to $\mathsf{Sub}(X)$.

Theorem 5.2.4.2. *$(\overline{(\cdot)}^{\mathbb{X}}, \eta^{\mathbb{X}}, \mu^{\mathbb{X}})$ is a monad on $\mathsf{Sub}(X)$.*

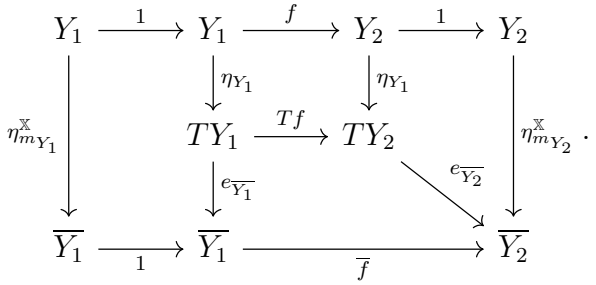
Proof. On the one hand, we have to establish the naturality of $\eta^{\mathbb{X}}$ and $\mu^{\mathbb{X}}$, that is, the identities

$$\begin{aligned} \overline{f} \circ \eta^{\mathbb{X}}_{m_{Y_1}} &= \eta^{\mathbb{X}}_{m_{Y_2}} \circ f \\ \overline{f} \circ \mu^{\mathbb{X}}_{m_{Y_1}} &= \mu^{\mathbb{X}}_{m_{Y_2}} \circ \overline{\overline{f}} \end{aligned} \tag{5.4}$$

for any subobject homomorphism $f : m_{Y_1} \to m_{Y_2}$. On the other hand, we need to establish the unitality and associativity laws:

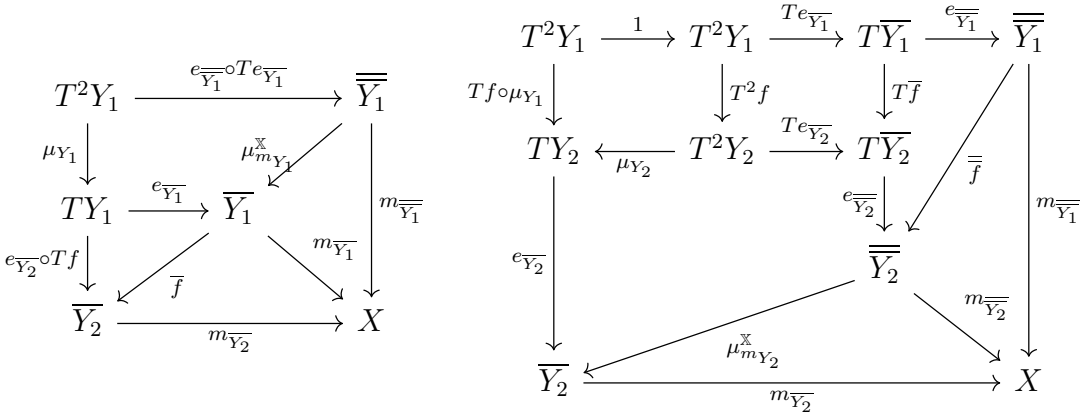
$$\begin{aligned} \mu^{\mathbb{X}}_{m_Y} \circ \eta^{\mathbb{X}}_{m_{\overline{Y}}} &= \mathsf{id}_{\overline{Y}} = \mu^{\mathbb{X}}_{m_Y} \circ \overline{\eta^{\mathbb{X}}_{m_Y}} \\ \mu^{\mathbb{X}}_{m_Y} \circ \mu^{\mathbb{X}}_{m_{\overline{Y}}} &= \mu^{\mathbb{X}}_{m_Y} \circ \overline{\mu^{\mathbb{X}}_{m_Y}}. \end{aligned} \tag{5.5}$$

The first equation of (5.4) follows from the commutativity of the diagram below:

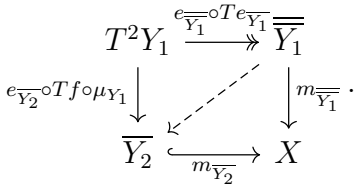


For the second equation of (5.4) we observe that since the following two diagrams

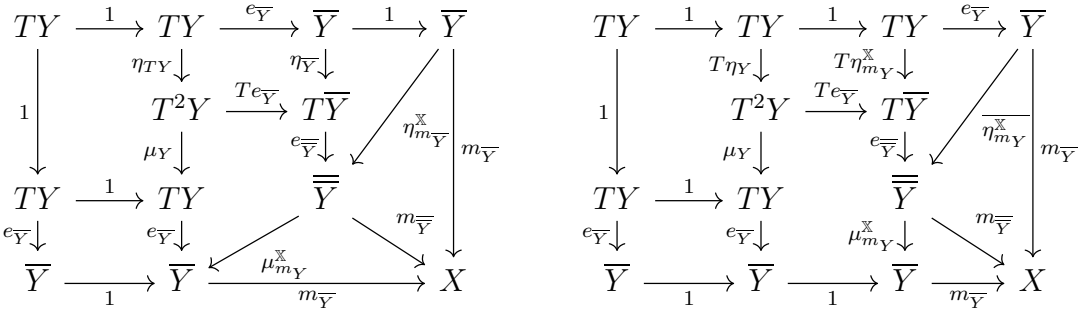
commute



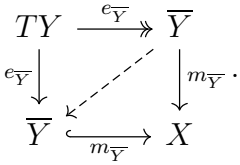
both $\overline{f} \circ \mu^{\mathbb{X}}_{m_{Y_1}}$ and $\mu^{\mathbb{X}}_{m_{Y_2}} \circ \overline{\overline{f}}$ are solutions to the unique diagonal below:



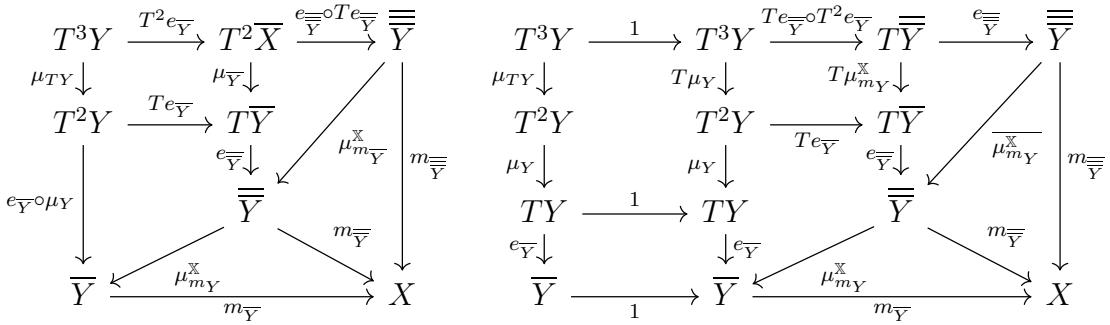
For the first equation of (5.5) we observe that since the following diagrams commute



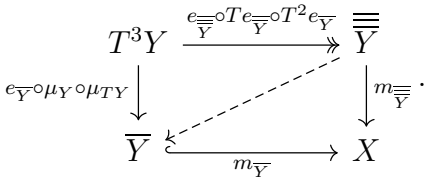
all three morphisms $\mu^{\mathbb{X}}_{m_Y} \circ \eta^{\mathbb{X}}_{m_{\overline{Y}}}$, $\mathsf{id}_{\overline{Y}}$ and $\mu^{\mathbb{X}}_{m_Y} \circ \overline{\eta^{\mathbb{X}}_{m_Y}}$ are solutions to the unique diagonal:



Similarly, for the second equation of (5.5) we note that since the following two diagrams commute



both morphisms $\mu^{\mathbb{X}}_{m_Y} \circ \mu^{\mathbb{X}}_{m_{\overline{Y}}}$ and $\mu^{\mathbb{X}}_{m_Y} \circ \overline{\mu^{\mathbb{X}}_{m_Y}}$ are solutions to the unique diagonal below:



□

As before, we exemplify Theorem 5.2.4.2 for the free vector space monad on the category of sets and functions with its canonical factorisation system.

Example 5.2.4.3. We have previously seen that the closure of a subobject m_Y of a vector space $\mathbb{V}$ consists of formal linear combinations that are considered equivalent, if their interpretation in $\mathbb{V}$ via m_Y coincides. By construction (cf. (5.3)), the monad unit $\eta^{\mathbb{V}}_{m_Y}$ maps an element $y \in Y$ to the equivalence class $[1 \cdot y]$ in $\overline{Y}$. One further verifies that the multiplication $\mu^{\mathbb{V}}_{m_Y}$ assigns to the element $[\sum_{[\varphi]} \lambda_{[\varphi]} \cdot [\varphi]]$ in $\overline{\overline{Y}}$ the

element $[\sum_x(\sum_{[\varphi]} \lambda_{[\varphi]} \cdot \varphi(y)) \cdot y]$ in $\overline{Y}$. If Y is a subset of $\mathbb{V}$, that is, $m_Y(y) = y$ for all $y \in Y$, the multiplication thus flattens vectors of vectors in the usual way.

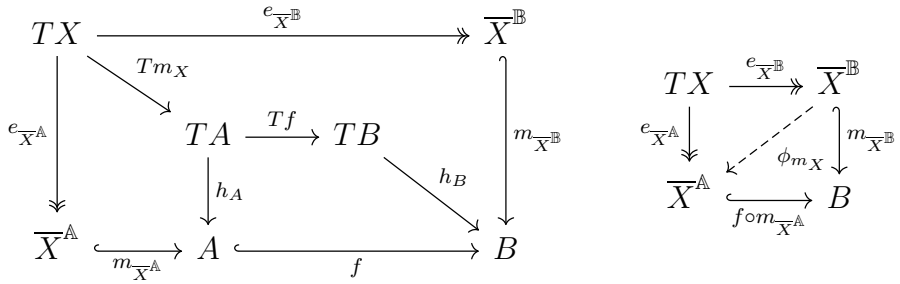
We will now show that the mapping of an algebra $\mathbb{X}$ to the monad $\overline{(\cdot)}^{\mathbb{X}}$ in Theorem 5.2.4.2 extends to algebra homomorphisms. To this end, for any algebra homomorphism $f : \mathbb{A} \to \mathbb{B}$ in $\mathscr{M}$, let $f_* : \mathsf{Sub}(A) \to \mathsf{Sub}(B)$ be the induced functor defined by $f_*(m_X) = f \circ m_X$ and $f_*(g) = g$. The result below shows that f_* can be extended to a morphism between monads.

Lemma 5.2.4.4. *For any $f : \mathbb{A} \to \mathbb{B} \in \mathscr{M}$, there exists a monad morphism $(f_*, \alpha) : (\mathsf{Sub}(A), \overline{(\cdot)}^{\mathbb{A}}) \to (\mathsf{Sub}(B), \overline{(\cdot)}^{\mathbb{B}})$.*

Proof. We need to define a natural transformation $\alpha : \overline{(\cdot)}^{\mathbb{B}} \circ f_* \Rightarrow f_* \circ \overline{(\cdot)}^{\mathbb{A}}$ between functors of type $\mathsf{Sub}(A) \to \mathsf{Sub}(B)$. That is, for any subobject $m_X : X \to A$, we require a homomorphism

$$\alpha_{m_X} : m_{\overline{X}^{\mathbb{B}}} \to f \circ m_{\overline{X}^{\mathbb{A}}}$$

between subobjects of B. Since factorisations are unique up to unique isomorphism, and the diagram on the left below commutes



there exists a unique homomorphism $\phi_{m_X} : m_{\overline{X}^{\mathbb{B}}} \to f \circ m_{\overline{X}^{\mathbb{A}}}$ of subobjects of B as indicated on the right above. We thus propose the definition

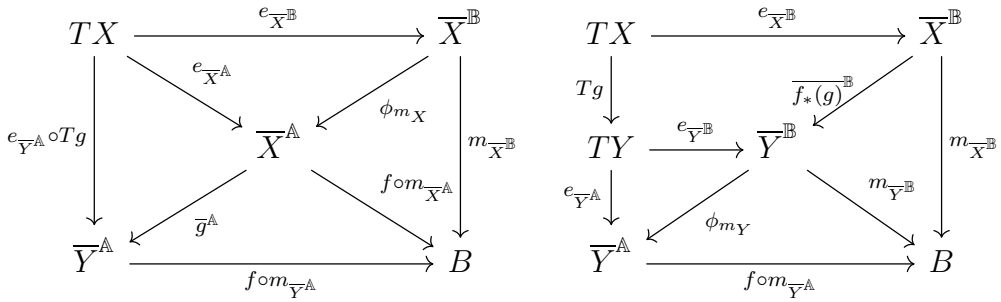
$$\alpha_{m_X} := \phi_{m_X}.$$

We begin by showing that the above proposal turns α into a *natural* transformation. Let $g : m_X \to m_Y$ be a morphism of subobjects of A and $f_*(g) = g : f_*(m_X) \to$

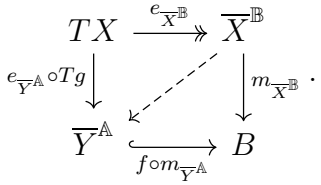
$f_*(m_Y)$ the induced morphism of subobjects of B. We need to prove the equality

$$\phi_{m_Y} \circ \overline{f_*(g)}^{\mathbb{B}} = \overline{g}^{\mathbb{A}} \circ \phi_{m_X}.$$

To this end, note that, as the two diagrams below commute



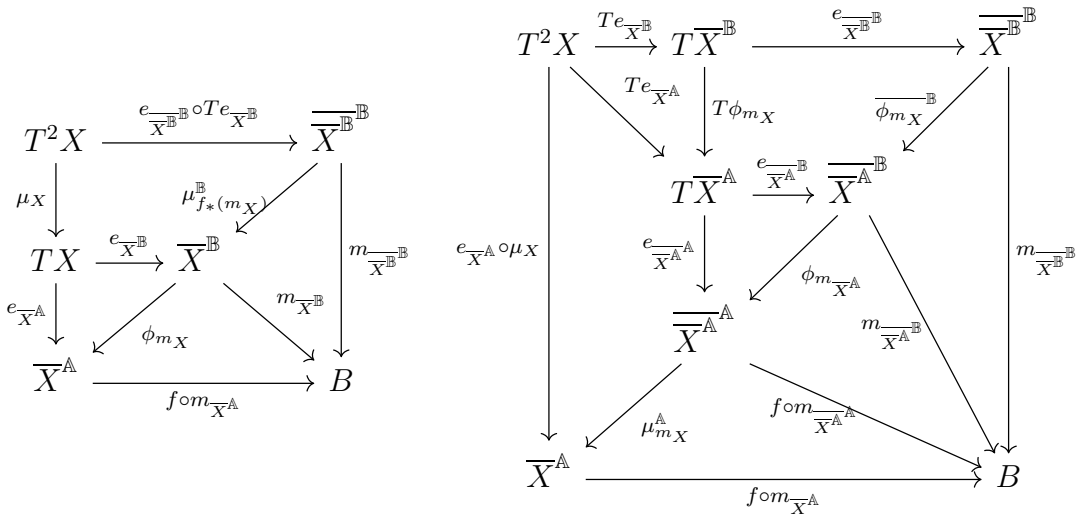
both $\phi_{m_Y} \circ \overline{f_*(g)}^{\mathbb{B}}$ and $\overline{g}^{\mathbb{A}} \circ \phi_{m_X}$ are solutions to the unique diagonal below:



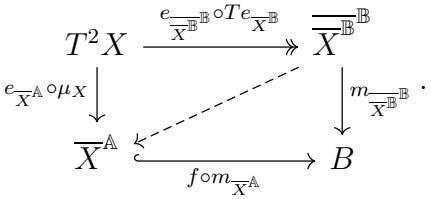
Finally, one verifies that the commutative diagrams turning (f_*, α) into a morphism between monads correspond to the two equations

$$\phi_{m_X} \circ \mu^{\mathbb{B}}_{f_*(m_X)} = \mu^{\mathbb{A}}_{m_X} \circ \phi_{m_{\overline{X}^{\mathbb{A}}}} \circ \overline{\phi_{m_X}}^{\mathbb{B}} \qquad \eta^{\mathbb{A}}_{m_X} = \phi_{m_X} \circ \eta^{\mathbb{B}}_{f_*(m_X)}.$$

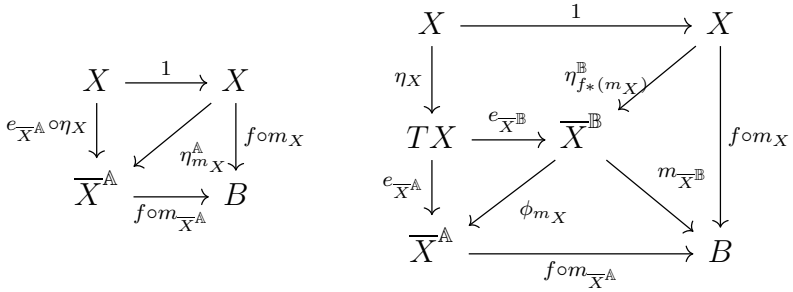
For the first equation we observe that since the two diagrams below commute



both $\phi_{m_X} \circ \mu^{\mathbb{B}}_{f_*(m_X)}$ and $\mu^{\mathbb{A}}_{m_X} \circ \phi_{m_{\overline{X}^{\mathbb{A}}}} \circ \overline{\phi_{m_X}}^{\mathbb{B}}$ are solutions to the unique diagonal below:



For the second equation we observe that since the two diagrams below commute



both $\eta^{\mathbb{A}}_{m_X}$ and $\phi_{m_X} \circ \eta^{\mathbb{B}}_{f_*(m_X)}$ are solutions to the unique diagonal below:

$$\begin{array}{ccc} X & \xrightarrow{1} & X \\ {\scriptstyle e_{\overline{X}^{\mathbb{A}}} \circ \eta_X}\downarrow & \swarrow & \downarrow{\scriptstyle f \circ m_X} \\ \overline{X}^{\mathbb{A}} & \xrightarrow[f \circ m_{\overline{X}^{\mathbb{A}}}]{} & B \end{array}.$$

□

The next statement establishes that the canonical forgetful functor $U : \mathsf{Sub}(X) \to \mathscr{C}$ defined by $U(m_Y) = Y$ and $U(f) = f$ extends to a morphism between monads.

Lemma 5.2.4.5. *There exists a monad morphism* $(U, \alpha) : (\mathsf{Sub}(X), \overline{(\cdot)}^{\mathbb{X}}) \to (\mathscr{C}, T)$.

Proof. We propose the following definition:

$$\alpha : T \circ U \Rightarrow U \circ \overline{(\cdot)}^{\mathbb{X}} \qquad \alpha_{m_Y} := e_{\overline{Y}} : TY \to \overline{Y}.$$

From the definition of $\overline{(\cdot)}^{\mathbb{X}}$ on morphisms it follows that α is a natural transformation. The commutative diagrams turning (U, α) into a morphism between monads correspond to the following two equations:

$$\mu^{\mathbb{X}}_{m_{\overline{Y}}} \circ e_{\overline{\overline{Y}}} \circ T e_{\overline{Y}} = e_{\overline{Y}} \circ \mu_Y \qquad \eta^{\mathbb{X}}_{m_Y} = e_{\overline{Y}} \circ \eta_Y.$$

The equalities are satisfied by the definitions of $\eta^{\mathbb{X}}$ and $\mu^{\mathbb{X}}$, respectively. □

We conclude with the observation that the monad morphism defined in Lemma 5.2.4.4 commutes with the monad morphisms defined in Lemma 5.2.4.5.

Lemma 5.2.4.6. *For any algebra homomorphism* $f : \mathbb{A} \to \mathbb{B} \in \mathscr{M}$*, the following diagram commutes:*

$$\begin{array}{ccc} (\mathsf{Sub}(A), \overline{(\cdot)}^{\mathbb{A}}) & \xrightarrow{(f_*, \alpha_f)} & (\mathsf{Sub}(B), \overline{(\cdot)}^{\mathbb{B}}) \\ & {\scriptstyle (U_{\mathbb{A}}, \alpha_{\mathbb{A}})} \searrow \quad \swarrow {\scriptstyle (U_{\mathbb{B}}, \alpha_{\mathbb{B}})} & \\ & (\mathscr{C}, T) & \end{array}$$

Proof. We have to show that the monad morphism $(U_{\mathbb{B}} \circ f_*, \beta) : (\mathsf{Sub}(A), \overline{(\cdot)}^{\mathbb{A}}) \to (\mathscr{C}, T)$ with the natural transformation

$$\beta = T \circ (U_{\mathbb{B}} \circ f_*) \overset{\alpha_{\mathbb{B}} \circ f_*}{\Longrightarrow} U_{\mathbb{B}} \circ \overline{(\cdot)}^{\mathbb{B}} \circ f_* \overset{U_{\mathbb{B}} \circ \alpha_f}{\Longrightarrow} (U_{\mathbb{B}} \circ f_*) \circ \overline{(\cdot)}^{\mathbb{A}}$$

given on a subobject $m_Y : Y \to A$ in $\mathsf{Sub}(A)$ by the morphism

$$\beta_{m_Y} : TY \overset{e_{\overline{Y}^{\mathbb{B}}}}{\longrightarrow} \overline{Y}^{\mathbb{B}} \overset{\phi_{m_Y}}{\longrightarrow} \overline{Y}^{\mathbb{A}}$$

coincides with the morphism $(U_{\mathbb{A}}, \alpha_{\mathbb{A}}) : (\mathsf{Sub}(A), \overline{(\cdot)}^{\mathbb{A}}) \to (\mathscr{C}, T)$ with $(\alpha_{\mathbb{A}})_{m_Y} = e_{\overline{Y}^{\mathbb{A}}}$.

The equality $U_{\mathbb{B}} \circ f_* = U_{\mathbb{A}}$ is immediate from the involved definitions. The identity $\phi_{m_Y} \circ e_{\overline{Y}^{\mathbb{B}}} = e_{\overline{Y}^{\mathbb{A}}}$ follows from the definition of ϕ_{m_Y} as unique diagonal. □

5.2.5 Closing an Image

In this section we investigate the closure of a particular class of subobjects: the ones that arise by taking the image of a morphism. We then show that deriving a minimal bialgebra from a minimal coalgebra by equipping the latter with additional algebraic structure can be realized as the closure of a subobject in this class.

We assume a category $\mathscr{C}$ with a factorisation system $(\mathscr{E}, \mathscr{M})$ and a monad T on $\mathscr{C}$ that preserves $\mathscr{E}$. Suppose that $\mathbb{X} = (X, h_X)$ is a T-algebra and $f : Y \to X$ a morphism in $\mathscr{C}$. On the one hand, there exists a factorisation of f in $\mathscr{C}$:

$$f = Y \overset{e_{\mathsf{im}(f)}}{\twoheadrightarrow} \mathsf{im}(f) \overset{m_{\mathsf{im}(f)}}{\hookrightarrow} X.$$

On the other hand, there exists a factorisation of the lifing $f^\sharp = h_X \circ Tf$ in the category of Eilenberg-Moore algebras $\mathscr{C}^T$:

$$f^\sharp = (TY, \mu_Y) \overset{e_{\mathsf{im}(f^\sharp)}}{\twoheadrightarrow} (\mathsf{im}(f^\sharp), h_{\mathsf{im}(f^\sharp)}) \overset{m_{\mathsf{im}(f^\sharp)}}{\hookrightarrow} (X, h_X).$$

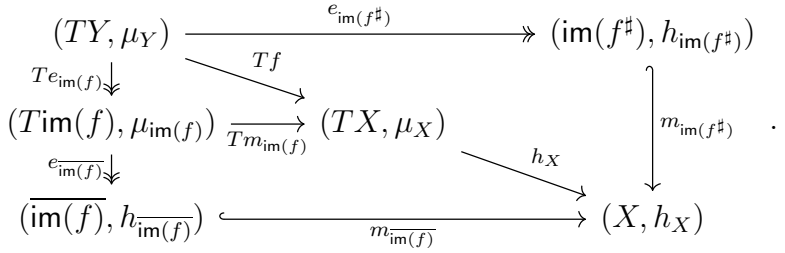
The next result shows that, up to isomorphism, the closure of the subobject $m_{\mathsf{im}(f)}$ with respect to the algebra $\mathbb{X}$ is given by the subobject $m_{\mathsf{im}(f^\sharp)}$.

Lemma 5.2.5.1. $\overline{m_{\mathsf{im}(f)}}^{\mathbb{X}} = m_{\mathsf{im}(f^\sharp)}$ *in* $\mathsf{Sub}(\mathbb{X})$.

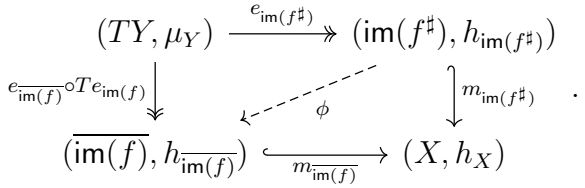
Proof. Using the factorisation of $(m_{\mathsf{im}(f)})^\sharp = h_X \circ T m_{\mathsf{im}(f)}$ in $\mathscr{C}^T$,

$$(m_{\mathsf{im}(f)})^\sharp = (T\mathsf{im}(f), \mu_{\mathsf{im}(f)}) \overset{e_{\overline{\mathsf{im}(f)}}}{\twoheadrightarrow} (\overline{\mathsf{im}(f)}, h_{\overline{\mathsf{im}(f)}}) \overset{m_{\overline{\mathsf{im}(f)}}}{\hookrightarrow} (X, h_X)$$

one easily verifies that the diagram below commutes:



Since factorisations are unique up to unique isomorphism, there thus exists a unique isomorphism $\phi : m_{\mathsf{im}(f^\sharp)} \simeq m_{\overline{\mathsf{im}(f)}}$ of subobjects of $\mathbb{X}$ as indicated below:



Since by definition $\overline{m_{\mathsf{im}(f)}}^{\mathbb{X}} \simeq m_{\overline{\mathsf{im}(f)}}$, this shows the claim. □

The next example uses Lemma 5.2.5.1 to show that deriving a minimal bialgebra from a minimal coalgebra can be realised as the closure of a subobject with respect to a monad of the type in Theorem 5.2.4.2.

Example 5.2.5.2 (Closure of Minimal Moore Automata)**.** Let F be the set endofunctor with $FX = B \times X^A$, for fixed sets A and B. Let T be a set monad, $h : TB \to B$ a T-algebra structure for B, and let $L : A^* \to B$ be a generalised language.

As F preserves monomorphisms, the canonical epi-mono factorisation system of the category of sets lifts to the category $\mathsf{Coalg}(F)$, which consists of unpointed Moore automata with input A and output B.

There exists a size-minimal Moore automaton M_L that accepts L. It can be recovered as the epi-mono factorisation of the final F-coalgebra homomorphism obs :

$A^* \to \Omega$, that is, $\mathsf{M}_L = m_{\mathsf{im}(\mathsf{obs})}$. In more detail: Ω is carried by B^{A^*}; obs satisfies $\mathsf{obs}(w)(v) = L(wv)$; and A^* is equipped with the F-coalgebra structure $\langle \varepsilon, \delta \rangle : A^* \to B \times (A^*)^A$ defined by $\varepsilon(w) = L(w)$ and $\delta(w)(a) = wa$ [63].

The algebra structure h induces a canonical[4] distributive law λ between T and F. One can show that λ-bialgebras are algebras over the monad T_λ on $\mathsf{Coalg}(F)$ defined by $T_\lambda(X,k) = (TX, \lambda_X \circ Tk)$ and $T_\lambda f = Tf$ [152]. One such T_λ-algebra is the final F-coalgebra Ω, when equipped with a canonical T-algebra structure [81, Prop. 3].

The functor T_λ preserves epimorphisms in the category $\mathsf{Coalg}(F)$, if T preserves epimorphisms in the category of sets. The latter is the case for every set endofunctor. By Theorem 5.2.4.2, there thus exists a well-defined monad $\overline{(\cdot)}$ on $\mathsf{Sub}(\Omega)$.

By construction, the minimal Moore automaton M_L lives in $\mathsf{Sub}(\Omega)$. Reviewing the constructions in [161] shows that the minimal λ-bialgebra $\mathbb{M}_L$ for L is given by the image of the lifting of obs, that is, $\mathbb{M}_L = m_{\mathsf{im}(\mathsf{obs}^\sharp)}$. From Lemma 5.2.5.1 it thus follows $\mathbb{M}_L = \overline{\mathsf{M}_L}$. Hence the minimal λ-bialgebra for L can be obtained from the minimal F-coalgebra for L by closing the latter with respect to the T_λ-algebra structure of Ω.

For an example of the monad unit, observe how the minimal coalgebra in Figure 4.3 embeds into the minimal bialgebra in Figure 4.6a.

The situation can be further generalised. We assume that i) $\mathscr{C}$ is a category with an $(\mathscr{E}, \mathscr{M})$-factorisation system; ii) λ is a distributive law between a monad T on $\mathscr{C}$ that preserves $\mathscr{E}$ and an endofunctor F on $\mathscr{C}$ that preserves $\mathscr{M}$; iii) $(\Omega, h_\Omega, k_\Omega)$ is a final λ-bialgebra.

Theorem 5.2.5.3. *There exists a functor* $\overline{(\cdot)} : \mathsf{Sub}(\Omega, k_\Omega) \to \mathsf{Sub}(\Omega, h_\Omega, k_\Omega)$ *yielding a monad on* $\mathsf{Sub}(\Omega, k_\Omega)$ *and satisfying* $\overline{m_{\mathsf{im}(\mathsf{obs}_{(X,k)})}} \cong m_{\mathsf{im}(\mathsf{obs}_{\mathsf{free}_T(X,k)})}$ *in* $\mathsf{Sub}(\Omega, h_\Omega, k_\Omega)$*, for any* F*-coalgebra* (X,k)*.*

Proof. As F preserves $\mathscr{M}$, the $(\mathscr{E}, \mathscr{M})$-factorisation system of $\mathscr{C}$ lifts to $\mathsf{Coalg}(F)$. The category of λ-bialgebras is isomorphic to the category of algebras over the monad T_λ on $\mathsf{Coalg}(F)$ defined by $T_\lambda(X,k) = (TX, \lambda_X \circ Tk)$ and $T_\lambda f = Tf$ [152]. The functor

[4] Given an algebra $h : TB \to B$ for a set monad T, one can define a distributive law λ between T and F with $FX = B \times X^A$ by $\lambda_X := (h \times \mathsf{st}) \circ \langle T\pi_1, T\pi_2 \rangle : TFX \to FTX$ [75]. (We write st for the usual strength function $\mathsf{st} : T(X^A) \to (TX)^A$ defined by $\mathsf{st}(U)(a) = T(\mathsf{ev}_a)(U)$, where $\mathsf{ev}_a(f) = f(a)$.)

T_λ preserves the $\mathscr{E}$-part of the lifted factorisation system of $\mathsf{Coalg}(F)$, if T preserves the $\mathscr{E}$-part of the factorisation system of $\mathscr{C}$. In consequence, the factorisation system of $\mathsf{Coalg}(F)$ thus lifts to $\mathsf{Bialg}(\lambda)$. By Proposition 5.2.3.1 and Theorem 5.2.4.2 it follows that there exists a functor $\overline{(\cdot)} : \mathsf{Sub}(\Omega, k_\Omega) \to \mathsf{Sub}(\Omega, h_\Omega, k_\Omega)$ that yields a monad on $\mathsf{Sub}(\Omega, k_\Omega)$. Since $(\Omega, h_\Omega, k_\Omega)$ is a final λ-bialgebra, (Ω, k_Ω) is a final F-coalgebra. By Lemma 5.2.5.1 we have the equality $\overline{m_{\mathsf{im}(\mathsf{obs}_{(X,k)})}} = m_{\mathsf{im}(\mathsf{obs}^\sharp_{(X,k)})}$ in $\mathsf{Sub}(\Omega, h_\Omega, k_\Omega)$, where $\mathsf{obs}^\sharp_{(X,k)} = h_\Omega \circ T_\lambda(\mathsf{obs}_{(X,k)})$ is of type $\mathsf{free}_T(X,k) = (TX, \mu_X, \lambda_X \circ Tk) \to (\Omega, h_\Omega, k_\Omega)$. By uniqueness it follows $\mathsf{obs}^\sharp_{(X,k)} = \mathsf{obs}_{\mathsf{free}_T(X,k)}$, which proves the claim. □

To recover Example 5.2.5.2 as a special case of Theorem 5.2.5.3, one instantiates the latter for F with $FX = B \times X^A$ and the canonical F-coalgebra with carrier A^*.

Finally, using analogous functors to the ones present in (5.2), we observe that, as a consequence of Lemma 5.2.5.1, the diagram below commutes:

$$\begin{array}{ccc} \mathscr{C}/X & \longrightarrow & \mathscr{C}^T/\mathbb{X} \\ \downarrow & & \downarrow \\ \mathsf{Sub}(X) & \xrightarrow{\overline{(\cdot)}^{\mathbb{X}}} & \mathsf{Sub}(\mathbb{X}) \end{array} .$$

5.3 Step 2: Generators and Bases

One of the central concepts of linear algebra is the notion of a basis for a vector space: a subset of a vector space is called a basis for the former if every vector can be uniquely written as a finite linear combination of basis elements. Part of the importance of bases stems from the convenient consequences that follow from their existence. For example, linear transformations between vector spaces admit matrix representations relative to pairs of bases [96], which can be used for efficient calculations. The idea of a basis however is not restricted to the theory of vector spaces: other algebraic theories have analogous notions of bases (and generators, by waiving the uniqueness constraint), for instance modules, semi-lattices, Boolean algebras, convex sets, and many more. In fact, the theory of bases for vector spaces is special only in the sense that every vector space admits a basis, which is not the case for e.g. modules.

In this section, we use the compact category-theoretical abstraction of generators and bases given in Definition 4.4.0.1 to lift results from one theory to the others. For example, one may wonder if there exists a matrix representation theory for convex sets that is analogous to the one of vector spaces.

5.3.1 Categorification

This section introduces morphisms between algebras with a generator or a basis.

Definition 5.3.1.1. The category $\mathsf{GAlg}(T)$ of algebras with a generator over a monad T is defined as follows:

- Objects are pairs $(\mathbb{X}_\alpha, \alpha)$, where $\mathbb{X}_\alpha = (X_\alpha, h_\alpha)$ is a T-algebra with generator $\alpha = (Y_\alpha, i_\alpha, d_\alpha)$.
- A morphism $(f, p) : (\mathbb{X}_\alpha, \alpha) \to (\mathbb{X}_\beta, \beta)$ consists of a T-algebra homomorphism $f : \mathbb{X}_\alpha \to \mathbb{X}_\beta$ and a Kleisli-morphism $p : Y_\alpha \to TY_\beta$, such that the diagram below commutes:

$$\begin{array}{ccccc} X_\alpha & \xrightarrow{d_\alpha} & TY_\alpha & \xrightarrow{i_\alpha^\sharp} & X_\alpha \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle p^\sharp} & & \downarrow{\scriptstyle f} \\ X_\beta & \xrightarrow{d_\beta} & TY_\beta & \xrightarrow{i_\beta^\sharp} & X_\beta \end{array} . \tag{5.6}$$

  Given $(f, p) : (\mathbb{X}_\alpha, \alpha) \to (\mathbb{X}_\beta, \beta)$ and $(g, q) : (\mathbb{X}_\beta, \beta) \to (\mathbb{X}_\gamma, \gamma)$, their composition is defined componentwise as $(g, q) \circ (f, p) := (g \circ f, q \cdot p)$, where $q \cdot p := \mu_{Y_\gamma} \circ Tq \circ p$ denotes the usual Kleisli-composition.

The category $\mathsf{BAlg}(T)$ of algebras with a basis is defined as the obvious full subcategory of $\mathsf{GAlg}(T)$.

Let $F : \mathscr{C}^T \to \mathsf{GAlg}(T)$ be the functor with $F(\mathbb{X}) := (\mathbb{X}, (X, \mathsf{id}_X, \eta_X))$ and $F(f : \mathbb{X} \to \mathbb{Y}) := (f, \eta_Y \circ f)$, and $U : \mathsf{GAlg}(T) \to \mathscr{C}^T$ the forgetful functor defined as the projection on the first component. Then F and U are in an adjoint relation:

Lemma 5.3.1.2. $F \dashv U : \mathsf{GAlg}(T) \leftrightarrows \mathscr{C}^T$.

Proof. Since every algebra can be generated by itself, the definition for F is well-defined on objects. For morphisms, one easily establishes (5.6) from the naturality

of η, the monad law $\mu_Y \circ T\eta_Y = \mathsf{id}_{TY}$, and the commutativity of f with algebra structures. The compositionality of F follows analogously; preservation of identity is trivial. For the natural isomorphism

$$\mathsf{Hom}_{\mathsf{GAlg}(T)}(F(\mathbb{X}), (\mathbb{X}_\alpha, \alpha)) \simeq \mathsf{Hom}_{\mathscr{C}^T}(\mathbb{X}, U(\mathbb{X}_\alpha, \alpha))$$

we propose mapping (f, p) to f, and conversely, f to $(f, d_\alpha \circ f)$. The latter is well-defined since

$$(d_\alpha \circ f)^\sharp \circ \eta_X = d_\alpha \circ f \quad \text{and} \quad i_\alpha^\sharp \circ (d_\alpha \circ f)^\sharp = i_\alpha^\sharp \circ d_\alpha \circ f^\sharp = f^\sharp = f \circ (\mathsf{id}_X)^\sharp.$$

Composition in one of the directions trivially yields the identity; for the other direction we note that if (f, p) satisfies (5.6), then $p = p^\sharp \circ \eta_X = d_\alpha \circ f$. □

5.3.2 Products

In this section we show that, under certain assumptions, the monoidal product of a base category naturally extends to a monoidal product of algebras with bases in the base category. As a natural example we obtain the tensor-product of vector spaces with fixed bases.

We assume basic familiarity with monoidal categories. A monoidal monad T on a monoidal category $(\mathscr{C}, \otimes, I)$ is a monad which is equipped with natural transformations $T_{X,Y} : TX \otimes TY \to T(X \otimes Y)$ and $T_0 : I \to TI$, satisfying certain coherence conditions (see e.g. [138]). One can show that, given such additional data, the monoidal structure of $\mathscr{C}$ induces a monoidal category $(\mathscr{C}^T, \boxtimes, (TI, \mu_I))$, if the following two assumptions are given [138, Corollary 2.5.6]:

(A1) For any two algebras $\mathbb{X}_\alpha = (X_\alpha, h_\alpha)$ and $\mathbb{X}_\beta = (X_\beta, h_\beta)$ the coequaliser $q_{\mathbb{X}_\alpha, \mathbb{X}_\beta}$ of the algebra homomorphisms $T(h_\alpha \otimes h_\beta)$ and $\mu_{X_\alpha \otimes X_\beta} \circ T(T_{X_\alpha, X_\beta})$ of type $(T(TX_\alpha \otimes TX_\beta), \mu_{TX_\alpha \otimes TX_\beta}) \to (T(X_\alpha \otimes X_\beta), \mu_{X_\alpha \otimes X_\beta})$ exists (we denote its codomain by $\mathbb{X}_\alpha \boxtimes \mathbb{X}_\beta := (X_\alpha \boxtimes X_\beta, h_{\alpha \boxtimes \beta})$);

(A2) Left and right-tensoring with the induced functor $\boxtimes$ preserves reflexive coequaliser.

The two monoidal products $\otimes$ and $\boxtimes$ are related via the natural embedding $\iota_{\mathbb{X}_\alpha,\mathbb{X}_\beta} := q_{\mathbb{X}_\alpha,\mathbb{X}_\beta} \circ \eta_{X_\alpha \otimes X_\beta} : X_\alpha \otimes X_\beta \to X_\alpha \boxtimes X_\beta$. One can show that the product $(TY_\alpha, \mu_{Y_\alpha}) \boxtimes (TY_\beta, \mu_{Y_\beta})$ is given by $(T(Y_\alpha \otimes Y_\beta), \mu_{Y_\alpha \otimes Y_\beta})$ and the coequaliser $q_{(TY_\alpha,\mu_{Y_\alpha}),(TY_\beta,\mu_{Y_\beta})}$ by $\mu_{Y_\alpha \otimes Y_\beta} \circ T(T_{Y_\alpha,Y_\beta})$ [138].

With the previous remarks in mind, we are able to claim the following.

Lemma 5.3.2.1. *Let T be a monoidal monad on $(\mathscr{C}, \otimes, I)$ such that (A1) and (A2) are satisfied. Let $\alpha = (Y_\alpha, i_\alpha, d_\alpha)$ and $\beta = (Y_\beta, i_\beta, d_\beta)$ be generators (bases) for T-algebras $\mathbb{X}_\alpha$ and $\mathbb{X}_\beta$. Then $\alpha \boxtimes \beta = (Y_\alpha \otimes Y_\beta, \iota_{\mathbb{X}_\alpha,\mathbb{X}_\beta} \circ (i_\alpha \otimes i_\beta), (d_\alpha \boxtimes d_\beta))$ is a generator (basis) for the T-algebra $\mathbb{X}_\alpha \boxtimes \mathbb{X}_\beta$.*

Proof. First, we calculate

$$
\begin{aligned}
& h_\alpha \boxtimes h_\beta \\
= \; & (\mathsf{id} = \mu_{X_\alpha \otimes X_\beta} \circ T(\eta_{X_\alpha \otimes X_\beta})) \\
& (h_\alpha \boxtimes h_\beta) \circ \mu_{X_\alpha \otimes X_\beta} \circ T(\eta_{X_\alpha \otimes X_\beta}) \\
= \; & (q_{\mathbb{X}_\alpha,\mathbb{X}_\beta} = h_\alpha \boxtimes h_\beta \text{ [138]}) \\
& q_{\mathbb{X}_\alpha,\mathbb{X}_\beta} \circ \mu_{X_\alpha \otimes X_\beta} \circ T(\eta_{X_\alpha \otimes X_\beta}) \qquad (5.7) \\
= \; & (q_{\mathbb{X}_\alpha,\mathbb{X}_\beta} \text{ is algebra homomorphism}) \\
& h_{\alpha \boxtimes \beta} \circ T(q_{\mathbb{X}_\alpha,\mathbb{X}_\beta}) \circ T(\eta_{X_\alpha \otimes X_\beta}) \\
= \; & (\text{Definition of } \iota_{\mathbb{X}_\alpha,\mathbb{X}_\beta}) \\
& h_{\alpha \boxtimes \beta} \circ T(\iota_{\mathbb{X}_\alpha,\mathbb{X}_\beta}).
\end{aligned}
$$

If α and β are generators, it thus follows

$$
\begin{aligned}
& h_{\alpha \boxtimes \beta} \circ T(\iota_{\mathbb{X}_\alpha,\mathbb{X}_\beta}) \circ T(i_\alpha \otimes i_\beta) \circ (d_\alpha \boxtimes d_\beta) \\
= \; & (5.7) \\
& (h_\alpha \boxtimes h_\beta) \circ T(i_\alpha \otimes i_\beta) \circ (d_\alpha \boxtimes d_\beta) \\
= \; & (T(f \otimes g) = Tf \boxtimes Tg \text{ [138]}) \\
& (h_\alpha \boxtimes h_\beta) \circ (T(i_\alpha) \boxtimes T(i_\beta)) \circ (d_\alpha \boxtimes d_\beta) \\
= \; & (\boxtimes \text{ is functorial}) \\
& (h_\alpha \circ T(i_\alpha) \circ d_\alpha) \boxtimes (h_\beta \circ T(i_\beta) \circ d_\beta)
\end{aligned}
$$

$$
\begin{aligned}
&= (\alpha, \beta \text{ are generators}) \\
&\quad \mathsf{id}_{X_\alpha} \boxtimes \mathsf{id}_{X_\beta} \\
&= (\boxtimes \text{ is functorial}) \\
&\quad \mathsf{id}_{X_\alpha \boxtimes X_\beta}.
\end{aligned}
$$

The additional equality for the case in which α and β are bases follows analogously. □

Corollary 5.3.2.2. *Let T be a monoidal monad on $(\mathscr{C}, \otimes, I)$ such that (A1) and (A2) are satisfied. The definitions $(\mathbb{X}_\alpha, \alpha)\boxtimes(\mathbb{X}_\beta, \beta) := (\mathbb{X}_\alpha \boxtimes \mathbb{X}_\beta, \alpha \boxtimes \beta)$ and $(f, p)\boxtimes(g, q) := (f \boxtimes g, T_{Y_{\alpha'}, Y_{\beta'}} \circ (p \otimes q))$ yield monoidal structures with unit $((TI, \mu_I), (I, \eta_I, \mathsf{id}_{TI}))$ on $\mathsf{GAlg}(T)$ and $\mathsf{BAlg}(T)$.*

Proof. By Lemma 5.3.2.1 the construction is well-defined on objects. That it is well-defined on morphisms, i.e. the commutativity of (5.6), is a consequence of the equalities $Tf \boxtimes Tg = T(f \otimes g)$ and $q_{\mathbb{X}_\alpha, \mathbb{X}_\beta} = h_\alpha \boxtimes h_\beta$ [138], which imply $(T_{Y_\alpha, Y_\beta} \circ (p \otimes q))^\sharp = (\mu_{Y_{\alpha'}} \boxtimes \mu_{Y_{\beta'}}) \circ (Tp \boxtimes Tq)$. The natural isomorphisms underlying the monoidal structure for $\mathscr{C}^T$ can be extended to morphisms in $\mathsf{GAlg}(T)$ by associating canonical Kleisli-morphisms between generators as in (5.8). □

We conclude by instantiating above construction to the setting of vector spaces.

Example 5.3.2.3 (Tensor Product of Vector Spaces)**.** Recall the free $\mathbb{K}$-vector space monad $\mathcal{V}_\mathbb{K}$ from Examples 2.2.0.8. The category of sets is monoidal (in fact, cartesian) with respect to the cartesian product $\times$ and the singleton set $\{\star\}$. The monad $\mathcal{V}_\mathbb{K}$ is monoidal when equipped with the morphisms $(\mathcal{V}_\mathbb{K})_{X,Y}(\varphi, \psi)(x, y) := \varphi(x) \cdot \psi(y)$ and $(\mathcal{V}_\mathbb{K})_0(\star)(\star) := 1_\mathbb{K}$ [123]. The category of $\mathcal{V}_\mathbb{K}$-algebras is isomorphic to the category of $\mathbb{K}$-vector spaces, and satisfies (A1) and (A2). The monoidal structure induced by $\mathcal{V}_\mathbb{K}$ is the usual tensor product $\otimes$ of vector spaces with the unit field $\mathcal{V}_\mathbb{K}(\{\star\}) \simeq \mathbb{K}$. Lemma 5.3.2.1 captures the well-known fact that the dimension of the tensor product of two vector spaces equals the product of the dimensions of each vector space. The structure maps of the product generator map an element (y_α, y_β) to the

vector $i(y_\alpha) \otimes i(y_\beta)$, and a vector x to the function $(d_\alpha \otimes d_\beta)(x)$, where

$$\begin{aligned} d_\alpha \otimes d_\beta = \overline{d_\alpha \times d_\beta} : \mathbb{X}_\alpha \otimes \mathbb{X}_\beta &\to (\mathcal{V}_\mathbb{K}(Y_\alpha), \mu_{Y_\alpha}) \otimes (\mathcal{V}_\mathbb{K}(Y_\beta), \mu_{Y_\beta}) \\ &\simeq (\mathcal{V}_\mathbb{K}(Y_\alpha \times Y_\beta), \mu_{Y_\alpha \otimes Y_\beta}) \end{aligned}$$

is the unique linear extension of the bilinear map defined by

$$(d_\alpha \times d_\beta)(x_\alpha, x_\beta)(y_\alpha, y_\beta) = d_\alpha(x_\alpha)(y_\alpha) \cdot d_\beta(x_\beta)(y_\beta).$$

5.3.3 Kleisli Representation Theory

In this section we use our category-theoretical definition of a basis to derive a representation theory for homomorphisms between algebras over monads that is analogous to the matrix representation theory for linear transformations between vector spaces.

In more detail, recall that a linear transformation $L : V \to W$ between k-vector spaces with finite bases $\alpha = \{v_1, ..., v_n\}$ and $\beta = \{w_1, ..., w_m\}$, respectively, admits a matrix representation $L_{\alpha\beta} \in \mathsf{Mat}_k(m, n)$ with $L(v_j) = \sum_i (L_{\alpha\beta})_{i,j} w_i$, such that for any vector v in V the coordinate vectors $L(v)_\beta \in k^m$ and $v_\alpha \in k^n$ satisfy the equality $L(v)_\beta = L_{\alpha\beta} v_\alpha$. A great amount of linear algebra is concerned with finding bases such that the corresponding matrix representation is in an efficient shape, for instance diagonalised. The following definitions generalise the situation by substituting Kleisli morphisms for matrices.

Definition 5.3.3.1. Let $\alpha = (Y_\alpha, i_\alpha, d_\alpha)$ and $\beta = (Y_\beta, i_\beta, d_\beta)$ be bases for T-algebras $\mathbb{X}_\alpha = (X_\alpha, h_\alpha)$ and $\mathbb{X}_\beta = (X_\beta, h_\beta)$, respectively. The basis representation $f_{\alpha\beta}$ of a T-algebra homomorphism $f : \mathbb{X}_\alpha \to \mathbb{X}_\beta$ with respect to α and β is defined by

$$f_{\alpha\beta} := Y_\alpha \xrightarrow{i_\alpha} X_\alpha \xrightarrow{f} X_\beta \xrightarrow{d_\beta} TY_\beta. \tag{5.8}$$

Conversely, the morphism $p^{\alpha\beta}$ associated with a Kleisli morphism $p : Y_\alpha \to TY_\beta$ with respect to α and β is defined by

$$p^{\alpha\beta} := X_\alpha \xrightarrow{d_\alpha} TY_\alpha \xrightarrow{Tp} T^2 Y_\beta \xrightarrow{\mu_{Y_\beta}} TY_\beta \xrightarrow{Ti_\beta} TX_\beta \xrightarrow{h_\beta} X_\beta. \tag{5.9}$$

$$A = L_{\alpha'\alpha'} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \quad L_{\alpha\alpha} = \begin{pmatrix} 3 & 2 \\ -5 & -3 \end{pmatrix}, \quad P = \begin{pmatrix} -1 & 1 \\ 2 & -1 \end{pmatrix}, \quad P^{-1} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}$$

Figure 5.2: The basis representation of the counter-clockwise rotation by 90 degree $L : \mathbb{R}^2 \to \mathbb{R}^2$, $L(v) = Av$ with respect to $\alpha = \{(1,2),(1,1)\}$ and $\alpha' = \{(1,0),(0,1)\}$ satisfies $L_{\alpha'\alpha'} = P^{-1}L_{\alpha\alpha}P$

The morphism associated with a Kleisli morphism should be understood as the linear transformation between vector spaces induced by some matrix of the right type. The following result confirms this intuition.

Lemma 5.3.3.2. *The function* (5.9) *is a T-algebra homomorphism $p^{\alpha\beta} : \mathbb{X}_\alpha \to \mathbb{X}_\beta$.*

Proof. Using Lemma 4.4.0.6 we deduce the commutativity of the following diagram:

$$\begin{array}{ccccccccccc}
TX_\alpha & \xrightarrow{Td_\alpha} & T^2Y_\alpha & \xrightarrow{T^2p} & T^3Y_\beta & \xrightarrow{T\mu_{Y_\beta}} & T^2Y_\beta & \xrightarrow{T^2i_\beta} & T^2X_\beta & \xrightarrow{Th_\beta} & TX_\beta \\
\downarrow{\scriptstyle h_\alpha} & & \downarrow{\scriptstyle \mu_{Y_\alpha}} & & \downarrow{\scriptstyle \mu_{TY_\beta}} & & \downarrow{\scriptstyle \mu_{Y_\beta}} & & \downarrow{\scriptstyle \mu_{X_\beta}} & & \downarrow{\scriptstyle h_\beta} \\
X_\alpha & \xrightarrow{d_\alpha} & TY_\alpha & \xrightarrow{Tp} & T^2Y_\beta & \xrightarrow{\mu_{Y_\beta}} & TY_\beta & \xrightarrow{Ti_\beta} & TX_\beta & \xrightarrow{h_\beta} & X_\beta
\end{array}.$$

□

The next result establishes a generalisation of the observation that for fixed bases, constructing a matrix representation of a linear transformation on the one hand, and associating a linear transformation to a matrix of the right type on the other hand, are mutually inverse operations.

Lemma 5.3.3.3. *The operations* (5.8) *and* (5.9) *are mutually inverse.*

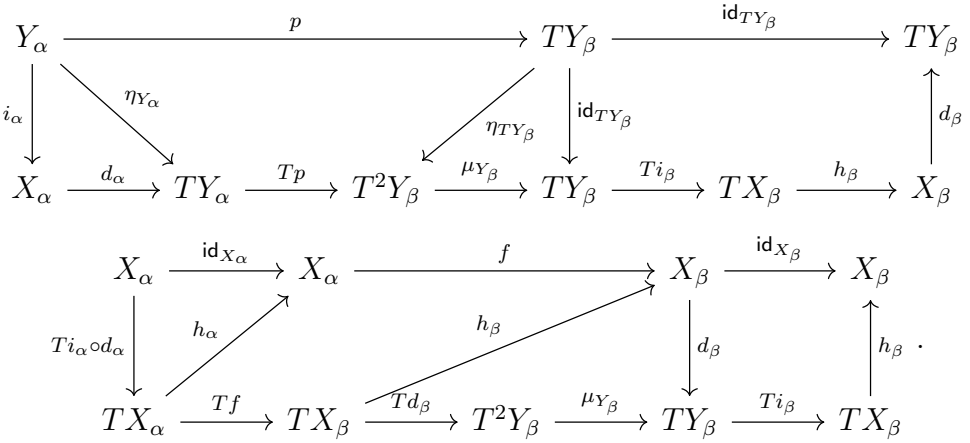
Proof. Essentially, the statement follows from the observation that, for bases, the functions involved in the composition below are isomorphisms:

$$\begin{aligned} \mathsf{Hom}_{\mathscr{C}^T}(\mathbb{X}_\alpha, \mathbb{X}_\beta) &\xrightarrow{(d_\beta)_* \circ (i_\alpha^\sharp)^*} \mathsf{Hom}_{\mathscr{C}^T}((TY_\alpha, \mu_{Y_\alpha}), (TY_\beta, \mu_{Y_\beta})) \\ &\xrightarrow{(\eta_{Y_\alpha})^*} \mathsf{Hom}_{\mathscr{C}_T}(Y_\alpha, Y_\beta). \end{aligned} \tag{5.10}$$

More concretely, the definitions imply:

$$(p^{\alpha\beta})_{\alpha\beta} = d_\beta \circ (h_\beta \circ Ti_\beta \circ \mu_{Y_\beta} \circ Tp \circ d_\alpha) \circ i_\alpha$$
$$(f_{\alpha\beta})^{\alpha\beta} = h_\beta \circ Ti_\beta \circ \mu_{Y_\beta} \circ T(d_\beta \circ f \circ i_\alpha) \circ d_\alpha.$$

Using Lemma 4.4.0.6 we deduce the commutativity of the diagrams below:



□

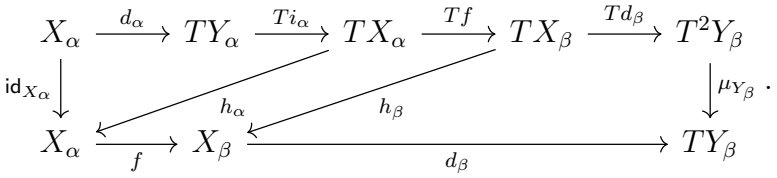
At the beginning of this section we recalled the soundness identity $L(v)_\beta = L_{\alpha\beta}v_\alpha$ for the matrix representation $L_{\alpha\beta}$ of a linear transformation L. The next result is a natural generalisation of this statement.

Lemma 5.3.3.4. *$f_{\alpha\beta}$ is the unique Kleisli-morphism such that $f_{\alpha\beta} \cdot d_\alpha = d_\beta \circ f$. Conversely, $p^{\alpha\beta}$ is the unique T-algebra homomorphism such that $p \cdot d_\alpha = d_\beta \circ p^{\alpha\beta}$.*

Proof. The definitions imply

$$f_{\alpha\beta} \cdot d_\alpha = \mu_{Y_\beta} \circ T(d_\beta \circ f \circ i_\alpha) \circ d_\alpha.$$

Using Lemma 4.4.0.6 we deduce the commutativity of the diagram below:



Since an equality of the type $p \cdot d_\alpha = d_\beta \circ f$ implies

$$p = \mu_{Y_\beta} \circ \eta_{TY_\beta} \circ p = \mu_{Y_\beta} \circ Tp \circ \eta_{Y_\alpha} = \mu_{Y_\beta} \circ Tp \circ d_\alpha \circ i_\alpha = d_\beta \circ f \circ i_\alpha = f_{\alpha\beta},$$

the morphism $f_{\alpha\beta}$ is moreover uniquely determined. For the second part of the claim we observe that by above and Lemma 5.3.3.3 it holds $p \cdot d_\alpha = (p^{\alpha\beta})_{\alpha\beta} \cdot d_\alpha = d_\beta \circ p^{\alpha\beta}$, and that an equality of the type $p \cdot d_\alpha = d_\beta \circ f$ implies $p^{\alpha\beta} = i_\beta^\sharp \circ (p \cdot d_\alpha) = i_\beta^\sharp \circ d_\beta \circ f = f$. □

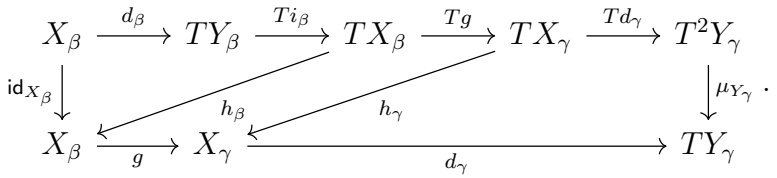
The next result establishes the compositionality of basis representations: the matrix representation of the composition of two linear transformations is given by the multiplication of the matrix representations of the individual linear transformations.

Lemma 5.3.3.5. $g_{\beta\gamma} \cdot f_{\alpha\beta} = (g \circ f)_{\alpha\gamma}$.

Proof. The definitions imply

$$\begin{aligned} g_{\beta\gamma} \cdot f_{\alpha\beta} &= \mu_{Y_\gamma} \circ T(d_\gamma \circ g \circ i_\beta) \circ d_\beta \circ f \circ i_\alpha \\ (g \circ f)_{\alpha\gamma} &= d_\gamma \circ (g \circ f) \circ i_\alpha. \end{aligned}$$

We delete common terms and use Lemma 4.4.0.6 to deduce the commutativity of the diagram below:



□

Similarly to the previous result, the next observation captures the compositionality of the operation that assigns to a Kleisli morphism its associated homomorphism.

Lemma 5.3.3.6. $q^{\beta\gamma} \circ p^{\alpha\beta} = (q \cdot p)^{\alpha\gamma}$.

Proof. The definitions imply

$$q^{\beta\gamma} \circ p^{\alpha\beta} = (h_\gamma \circ Ti_\gamma \circ \mu_{Y_\gamma} \circ Tq \circ d_\beta) \circ (h_\beta \circ Ti_\beta \circ \mu_{Y_\beta} \circ Tp \circ d_\alpha)$$
$$(q \cdot p)^{\alpha\gamma} = h_\gamma \circ Ti_\gamma \circ \mu_{Y_\gamma} \circ T\mu_{Y_\gamma} \circ T^2 q \circ Tp \circ d_\alpha.$$

By deleting common terms and using the equality $d_\beta \circ h_\beta \circ Ti_\beta = \mathsf{id}_{TY_\beta}$ it is thus sufficient to show

$$\mu_{Y_\gamma} \circ Tq \circ \mu_{Y_\beta} = \mu_{Y_\gamma} \circ T\mu_{Y_\gamma} \circ T^2 q.$$

Above equation follows from the commutativity of the diagram below:

$$\begin{array}{ccccc} T^2Y_\beta & \xrightarrow{\mu_{Y_\beta}} & TY_\beta & \xrightarrow{Tq} & T^2Y_\gamma \\ {\scriptstyle T^2q}\downarrow & \nearrow{\scriptstyle \mu_{TY_\gamma}} & & & \downarrow{\scriptstyle \mu_{Y_\gamma}} \\ T^3Y_\gamma & \xrightarrow[T\mu_{Y_\gamma}]{} & T^2Y_\gamma & \xrightarrow[\mu_{Y_\gamma}]{} & TY_\gamma \end{array}.$$

□

The previous statements may be summarised as functors between the following two categories, which arise from the usual Eilenberg-Moore and Kleisli categories.

Definition 5.3.3.7 ($\mathsf{Alg}_\mathsf{B}(T)$ and $\mathsf{Kl}_\mathsf{B}(T)$)**.** Let $\mathsf{Alg}_\mathsf{B}(T)$ be the category defined as follows:

- objects are given by pairs $(\mathbb{X}_\alpha, \alpha)$, where $\mathbb{X}_\alpha$ is a T-algebra with basis $\alpha = (Y_\alpha, i_\alpha, d_\alpha)$; and
- a morphism $f : (\mathbb{X}_\alpha, \alpha) \to (\mathbb{X}_\beta, \beta)$ consists of a T-algebra homomorphism $f : \mathbb{X}_\alpha \to \mathbb{X}_\beta$.

Similarly, let $\mathsf{Kl}_\mathsf{B}(T)$ be the category defined as follows:

- objects are given by pairs $(\mathbb{X}_\alpha, \alpha)$, where $\mathbb{X}_\alpha$ is a T-algebra with basis $\alpha = (Y_\alpha, i_\alpha, d_\alpha)$; and

- a morphism $p : (\mathbb{X}_\alpha, \alpha) \to (\mathbb{X}_\beta, \beta)$ consists of a morphism $p : Y_\alpha \to TY_\beta$; the composition is given by the usual Kleisli-composition.

Corollary 5.3.3.8. *There exist the following isomorphisms of categories:*

$$\mathsf{BAlg}(T) \simeq \mathsf{Alg}_{\mathsf{B}}(T) \simeq \mathsf{Kl}_{\mathsf{B}}(T).$$

Proof. For the first isomorphism we define a functor $F : \mathsf{BAlg}(T) \to \mathsf{Alg}_{\mathsf{B}}(T)$ by $F(\mathbb{X}_\alpha, \alpha) = (\mathbb{X}_\alpha, \alpha)$ and $F(f, p) = f$; and a functor $G : \mathsf{Alg}_{\mathsf{B}}(T) \to \mathsf{BAlg}(T)$ by $G(\mathbb{X}_\alpha, \alpha) = (\mathbb{X}_\alpha, \alpha)$ and $G(f) := (f, f_{\alpha\beta})$. The functoriality of G is a consequence of Lemma 5.3.3.5. The mutual invertibility of F and G is a consequence of Lemma 5.3.3.4. For the second isomorphism we define a functor $F : \mathsf{Alg}_{\mathsf{B}}(T) \to \mathsf{Kl}_{\mathsf{B}}(T)$ by $F(\mathbb{X}_\alpha, \alpha) = (\mathbb{X}_\alpha, \alpha)$ and $Ff = f_{\alpha\beta}$; and a functor $G : \mathsf{Kl}_{\mathsf{B}}(T) \to \mathsf{Alg}_{\mathsf{B}}(T)$ by $G(\mathbb{X}_\alpha, \alpha) = (\mathbb{X}_\alpha, \alpha)$ and $Gp = p^{\alpha\beta}$. The functoriality of F and G is a consequence of Lemma 5.3.3.5 and Lemma 5.3.3.6, respectively. Their mutual invertibility is a consequence Lemma 5.3.3.3. □

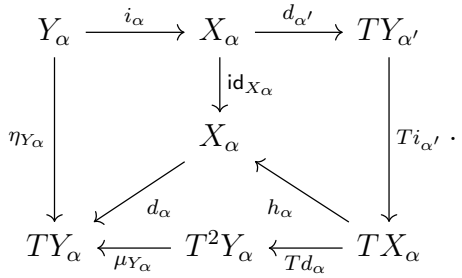
Assume we are given bases α, α' and β, β' for T-algebras (X_α, h_α) and (X_β, h_β), respectively. The following result clarifies how the two basis representations $f_{\alpha\beta}$ and $f_{\alpha'\beta'}$ are related.

Proposition 5.3.3.9. *There exist Kleisli isomorphisms p and q such that $f_{\alpha'\beta'} = q \cdot f_{\alpha\beta} \cdot p$.*

Proof. The Kleisli morphisms p and q and their respective candidates for inverses p^{-1} and q^{-1} are defined below

$$\begin{aligned} p &:= d_\alpha \circ i_{\alpha'} : Y_{\alpha'} \longrightarrow TY_\alpha & \qquad q &:= d_{\beta'} \circ i_\beta : Y_\beta \longrightarrow TY_{\beta'} \\ p^{-1} &:= d_{\alpha'} \circ i_\alpha : Y_\alpha \longrightarrow TY_{\alpha'} & \qquad q^{-1} &:= d_\beta \circ i_{\beta'} : Y_{\beta'} \longrightarrow TY_\beta. \end{aligned}$$

From Lemma 4.4.0.6 it follows that the diagram below commutes:

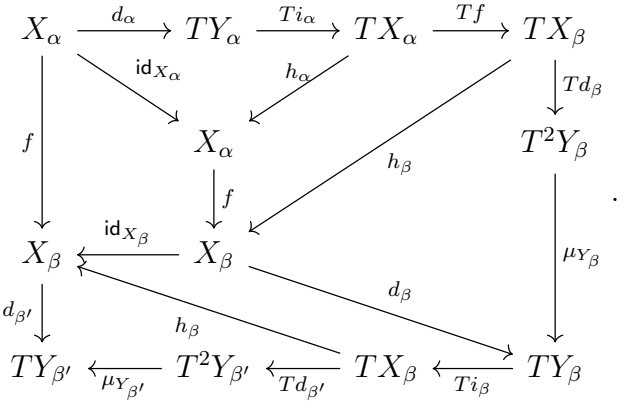


This shows that p^{-1} is a Kleisli right-inverse of p. A symmetric version of above diagram shows that p^{-1} is also a Kleisli left-inverse of p. Analogously it follows that q^{-1} is a Kleisli inverse of q.

The definitions further imply the equalities

$$q \cdot f_{\alpha\beta} \cdot p = \mu_{Y_{\beta'}} \circ T(d_{\beta'} \circ i_\beta) \circ \mu_{Y_\beta} \circ T(d_\beta \circ f \circ i_\alpha) \circ d_\alpha \circ i_{\alpha'}$$
$$f_{\alpha'\beta'} = d_{\beta'} \circ f \circ i_{\alpha'}.$$

We delete common terms and use Lemma 4.4.0.6 to establish the commutativity of the diagram below:



□

Above result simplifies if one restricts to an endomorphism: the basis representations are *similar*. This generalises the situation for vector spaces, cf. Figure 5.2.

Corollary 5.3.3.10. *There exists a Kleisli isomorphism p with Kleisli inverse p^{-1} such that $f_{\alpha'\alpha'} = p^{-1} \cdot f_{\alpha\alpha} \cdot p$.*

Proof. In Proposition 5.3.3.9 let $\beta = \alpha$ and $\beta' = \alpha'$. One verifies that in the corresponding proof the definitions of the morphisms p^{-1} and q coincide. □

5.3.4 Bases for Bialgebras

This subsection is concerned with generators and bases for a bialgebra. It is well-known [152] that an Eilenberg-Moore law λ between a monad T and an endofunctor F induces simultaneously

- a monad $T_\lambda = (T_\lambda, \mu, \eta)$ on $\mathsf{Coalg}(F)$ by $T_\lambda(X,k) = (TX, \lambda_X \circ Tk)$ and $T_\lambda f = Tf$; and
- an endofunctor F_λ on $\mathscr{C}^T$ by $F_\lambda(X,h) = (FX, Fh \circ \lambda_X)$ and $F_\lambda f = Ff$,

such that the algebras over T_λ, the coalgebras of F_λ, and λ-bialgebras coincide. We will consider generators and bases for T_λ-algebras, or equivalently, λ-bialgebras.

By Definition 4.4.0.1, a generator for a λ-bialgebra (X,h,k) consists of an F-coalgebra (Y,k_Y) and morphisms $i : Y \to X$ and $d : X \to TY$, such that the three diagrams on the left below commute:

$$\begin{array}{ccc} Y & \xrightarrow{i} & X \\ {\scriptstyle k_Y}\downarrow & & \downarrow{\scriptstyle k} \\ FY & \xrightarrow{Fi} & FX \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{d} & TY \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle \lambda_Y \circ Tk_Y} \\ FX & \xrightarrow{Fd} & FTY \end{array} \qquad \begin{array}{ccc} TY & \xrightarrow{Ti} & TX \\ {\scriptstyle d}\uparrow & & \downarrow{\scriptstyle h} \\ X & \xrightarrow{\mathsf{id}_X} & X \end{array} \qquad \begin{array}{ccc} TX & \xrightarrow{h} & X \\ {\scriptstyle Ti}\uparrow & & \downarrow{\scriptstyle d} \\ TY & \xrightarrow{\mathsf{id}_{TY}} & TY \end{array} . \tag{5.11}$$

A basis for a λ-bialgebra is a generator, such that in addition the diagram on the right above commutes.

It is easy to verify that by forgetting the F-coalgebra structure, every generator for a bialgebra in particular provides a generator for its underlying T-algebra. By Proposition 4.4.0.5 it thus follows that there exists a λ-bialgebra homomorphism $i^\sharp : \mathsf{exp}(Y, Fd \circ k \circ i) \to (X,h,k)$. The next result establishes that there exists a second equivalent free bialgebra with a different coalgebra structure.

Lemma 5.3.4.1. *Let (Y, k_Y, i, d) be a generator for (X, h, k). Then $i^\sharp : TY \to X$ is a λ-bialgebra homomorphism $i^\sharp : \mathsf{free}_T(Y, k_Y) \to (X, h, k)$.*

Proof. By definition we have $\mathsf{free}_T(Y, k_Y) = (TY, \mu_Y, \lambda_Y \circ Tk_Y)$. Clearly the lifting $i^\sharp = h \circ Ti$ is a T-algebra homomorphism $i^\sharp : (TY, \mu_Y) \to (X, h)$. It is an F-coalgebra homomorphism $i^\sharp : (TY, \lambda_Y \circ Tk_Y) \to (X, k)$ since the diagram below commutes:

$$\begin{array}{ccccc}
TY & \xrightarrow{Ti} & TX & \xrightarrow{h} & X \\
{\scriptstyle Tk_Y}\downarrow & & \downarrow{\scriptstyle Tk} & & \\
TFY & \xrightarrow{TFi} & TFX & & \downarrow{\scriptstyle k} \\
{\scriptstyle \lambda_Y}\downarrow & & \downarrow{\scriptstyle \lambda_X} & & \\
FTY & \xrightarrow{FTi} & FTX & \xrightarrow{Fh} & FX
\end{array} .$$

□

If one moves from generators to bases for bialgebras, both structures coincide.

Lemma 5.3.4.2. *Let (Y, k_Y, i, d) be a basis for (X, h, k), then $\mathsf{free}_T(Y, k_Y) = \mathsf{exp}_T(Y, Fd \circ k \circ i)$.*

Proof. Using Lemma 4.4.0.6 we establish the commutativity of the diagram below:

$$\begin{array}{ccccccccc}
TY & \xrightarrow{Tk_Y} & TFY & & \xrightarrow{\lambda_Y} & & FTY & \xrightarrow{\mathsf{id}_{FTY}} & FTY \\
 & & & & & & {\scriptstyle FTi}\downarrow & & \\
{\scriptstyle \mathsf{id}_{TY}}\downarrow & & \downarrow{\scriptstyle TFi} & & & & FTX & \xrightarrow{Fh} FX & \downarrow{\scriptstyle \mathsf{id}_{FTY}} \\
 & & & & & & {\scriptstyle FTd}\downarrow & \searrow{\scriptstyle Fd} & \\
TY & \xrightarrow{Ti} TX & \xrightarrow{Tk} TFX & \xrightarrow{TFd} & TFTY & \xrightarrow{\lambda_{TY}} & FT^2Y & \xrightarrow{F\mu_Y} & FTY
\end{array} .$$

□

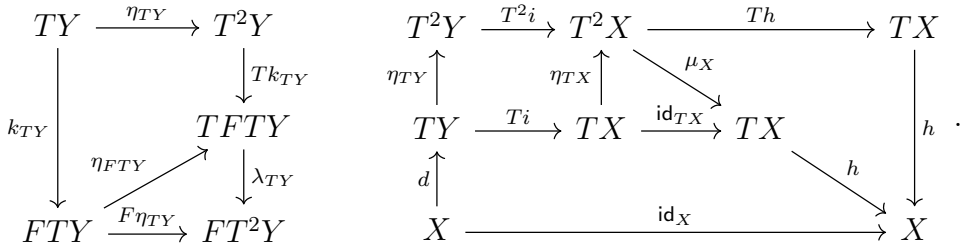
Example 5.3.4.3 (Canonical RFSA)**.** In Example 4.4.0.9 we considered the generator $(J(L), i, d)$ with $i(y) = y$ and $d(x) = \{y \in J(L) \mid y \leq x\}$ for the underlying algebraic part of the minimal pointed bialgebra (X, h, k) to recover the canonical RFSA for $L = (a + b)^* a$ as the coalgebra $Fd \circ k \circ i$. Figure 4.6 shows that the coalgebraic part k restricts to the join-irreducibles $J(L)$, suggesting $\alpha = (J(L), k, i, d)$

as a possible generator for the full bialgebra. However, as one easily verifies, the a-action on $[\{y\}]$ implies the non-commutativity of the second diagram on the left of (5.11). The issue can be fixed by modifying the definition of d by $d([\{y\}]) := \{[\{y\}]\}$. In consequence $\mathsf{free}_{\mathcal{P}}(J(L)), k)$ and $\mathsf{exp}_{\mathcal{P}}(J(L), Fd \circ k \circ i)$ coincide (even though the modification does not yield a basis).

We close this section by observing that a basis for the underlying algebra of a bialgebra is sufficient for constructing a generator for the full bialgebra.

Lemma 5.3.4.4. *Let (X, h, k) be a λ-bialgebra and (Y, i, d) a basis for the T-algebra (X, h). Then $(TY, F\mu_Y \circ \lambda_{TY} \circ T(Fd \circ k \circ i), h \circ Ti, \eta_{TY} \circ d)$ is a generator for (X, h, k).*

Proof. In the following we abbreviate $k_{TY} := F\mu_Y \circ \lambda_{TY} \circ T(Fd \circ k \circ i) : TY \to FTY$. By Proposition 4.4.0.5 the lifting $h \circ Ti$ is an F-coalgebra homomorphism $h \circ Ti : (TY, k_{TY}) \to (X, k)$. This shows the commutativity of the diagram on the left of (5.11). By Proposition 4.4.0.8 the morphism d is an F-coalgebra homomorphism in the reverse direction. Together with the commutativity of the diagram on the left below this implies the commutativity of the second diagram to the left of (5.11):



Similarly, the commutativity of third diagram to the left of (5.11) follows from the commutativity of the diagram on the right above. □

5.3.5 Bases as Coalgebras

In this section, we compare our approach with an alternative, coalgebraic, perspective on the generalisation of bases. More specifically, we are interested in the work of Jacobs [77], where a basis for an algebra over a monad is defined as a coalgebra for the comonad on the category of Eilenberg-Moore algebras induced by the free algebra

adjunction. Explicitly, a basis for a T-algebra (X, h), in the sense of [77], consists of a T-coalgebra (X, k) such that the following three diagrams commute:

$$\begin{array}{ccc} TX & \xrightarrow{Tk} & T^2X \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle \mu_X} \\ X & \xrightarrow{k} & TX \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{k} & TX \\ & {\scriptstyle \mathsf{id}_X}\searrow & \downarrow{\scriptstyle h} \\ & & X \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{k} & TX \\ {\scriptstyle k}\downarrow & & \downarrow{\scriptstyle T\eta_X} \\ TX & \xrightarrow{Tk} & T^2X \end{array} . \tag{5.12}$$

Next we show that a basis as in Definition 4.4.0.1 induces a basis as in [77].

Lemma 5.3.5.1. *Let (Y, i, d) be a basis for a T-algebra (X, h). Then (5.12) commutes for $k := Ti \circ d$.*

Proof. The commutativity of the diagram on the left of (5.12) follows from the naturality of μ and Lemma 4.4.0.6. The diagram in the middle of (5.12) commutes by the definition of a generator. The commutativity of the diagram on the right of (5.12) is again a consequence of Lemma 4.4.0.6:

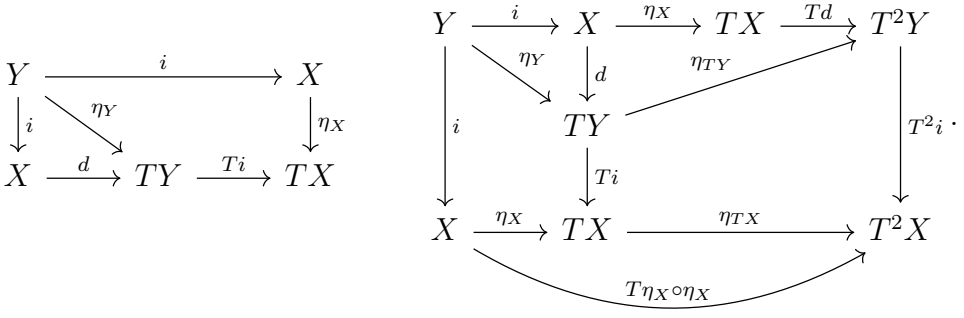
$$\begin{array}{ccccc} X & \xrightarrow{d} & TY & \xrightarrow{Ti} & TX \\ {\scriptstyle d}\downarrow & \nearrow{\scriptstyle \mathsf{id}_{TY}} & & & \\ TY & & & & \downarrow{\scriptstyle T\eta_X} \\ {\scriptstyle Ti}\downarrow & \searrow{\scriptstyle T\eta_Y} & & & \\ TX & \xrightarrow{Td} & T^2Y & \xrightarrow{T^2i} & T^2X \end{array} .$$

□

Conversely, assume (X, k) is a T-coalgebra structure satisfying (5.12) and $i_k : Y_k \to X$ an equaliser of k and η_X. If the underlying category is the category of sets and functions, the equaliser of any two functions exists. If Y_k is non-empty, one can show that the equaliser is preserved under T, that is, Ti_k is an equaliser of Tk and $T\eta_X$ [77]. By (5.12) we have $Tk \circ k = T\eta_X \circ k$. Thus there exists a unique morphism $d_k : X \to TY_k$ such that $Ti_k \circ d_k = k$, which can be shown to be the inverse of $h \circ Ti_k$ [77]. In other words, $G(X, k) := (Y_k, i_k, d_k)$ is a basis for (X, h) in the sense of Definition 4.4.0.1. In the following let $F(Y, i, d) := (X, Ti \circ d)$ for any basis of (X, h).

Lemma 5.3.5.2. *Let (Y, i, d) be a basis for a T-algebra (X, h) and $k := Ti \circ d$. Then $\eta_X \circ i = k \circ i$ and $Tk \circ (\eta_X \circ i) = T\eta_X \circ (\eta_X \circ i)$.*

Proof. The statement follows from Lemma 4.4.0.6:



□

Corollary 5.3.5.3. *Let $\alpha := (Y, i, d)$ be a basis for a set-based T-algebra (X, h) and $k := Ti \circ d$. Let $i_k : Y_k \to X$ be an equaliser of k and η_X, and Y_k non-empty, then $(\mathsf{id}_{(X,h)})_{\alpha,GF\alpha} : Y \to TY_k$ is the unique morphism ψ making the diagram below commute:*

$$Y \underset{\psi}{\overset{\eta_X \circ i}{\dashrightarrow}} TY_k \xrightarrow[Ti_k]{} TX \underset{T\eta_X}{\overset{Tk}{\rightrightarrows}} T^2X \ .$$

Proof. Since i_k is an equaliser of k and η_X, it follows from Lemma 5.3.5.2 that there exists a unique morphism $\varphi : Y \to Y_k$ such that $i_k \circ \varphi = i$. Since Y_k is non-empty, Ti_k is an equaliser of Tk and $T\eta_X$ [77]. It follows from Lemma 5.3.5.2 that there exists a unique morphism $\psi : Y \to TY_k$ such that $Ti_k \circ \psi = \eta_X \circ i$. It is not hard to see that $\psi = \eta_{Y_k} \circ \varphi$. The statement thus follows from $(\mathsf{id}_{(X,h)})_{\alpha,GF\alpha} = d_k \circ i = d_k \circ i_k \circ \varphi = \eta_{Y_k} \circ \varphi = \psi$. □

5.3.6 Signatures, Equations, and Finitary Monads

Most of the algebras over set monads one usually considers generators for constitute finitary varieties in the sense of universal algebra. In this section, we will briefly explore the consequences for generators that arise from this observation. The constructions are well-known; we include them for completeness.

Let Σ be a set, whose elements we think of as *operations*, and $\mathrm{ar} : \Sigma \to \mathbb{N}$ a function that assigns to an operation its *arity*. Any such *signature* induces a set endofunctor

H_Σ defined on a set as the coproduct $H_\Sigma X = \coprod_{\sigma\in\Sigma} X^{\mathrm{ar}(\sigma)}$, and consequently, a set monad $\mathbb{S}_\Sigma$ that assigns to a set V of variables the initial algebra $S_\Sigma V = \mu X.(V + H_\Sigma X)$, i.e. the set of Σ-terms generated by V (see e.g. [151]). One can show that the categories of H_Σ-algebras and $\mathbb{S}_\Sigma$-algebras are isomorphic. A $\mathbb{S}_\Sigma$-algebra $\mathbb{X}$ *satisfies a set of equations* $E \subseteq S_\Sigma V \times S_\Sigma V$, if for all $(s,t) \in E$ and valuations $v : V \to X$ it holds $v^\sharp(s) = v^\sharp(t)$, where $v^\sharp : (S_\Sigma V, \mu_V) \to \mathbb{X}$ is the unique extension of v to a $\mathbb{S}_\Sigma$-algebra homomorphism [5]. The set of $\mathbb{S}_\Sigma$-algebras that satisfy E is denoted by $\mathsf{Alg}(\Sigma, E)$. As one verifies, the forgetful functor $U : \mathsf{Alg}(\Sigma, E) \to \mathsf{Set}$ admits a left-adjoint $F : \mathsf{Set} \to \mathsf{Alg}(\Sigma, E)$, thus resulting in a set monad $T_{\Sigma,E}$ with underlying endofunctor $U \circ F$ that preserves directed colimits. The functor U can be shown to be monadic, that is, the comparison functor $K : \mathsf{Alg}(\Sigma, E) \to \mathsf{Set}^{T_{\Sigma,E}}$ is an isomorphism [105]. In other words, the category of Eilenberg-Moore algebras over $T_{\Sigma,E}$ and the finitary variety of algebras over Σ and E coincide. In fact, set monads preserving directed colimits (*finitary* monads [5]) and finitary varieties are in *bijection*.

The following result characterises generators for algebras over $T_{\Sigma,E}$. It can be seen as a unifying proof for observations analogous to the one in Example 4.4.0.2. For any Σ-term $t \in \mathbb{S}_\Sigma V$ over variables V, let $[\![t]\!]_E \in S_\Sigma V/\cong_E$ denote the equivalence class of t w.r.t. the smallest congruence relation $\cong_E$ on $S_\Sigma V$ generated by the equations E.

Lemma 5.3.6.1. *A morphism $i : Y \to X$ is part of a generator for a $T_{\Sigma,E}$-algebra $\mathbb{X}$ iff every element $x \in X$ can be expressed as a Σ-term in $i[Y]$ modulo E, that is, there is a term $d(x) \in S_\Sigma Y$ such that $i^\sharp([\![d(x)]\!]_E) = x$.*

Proof. Let $i : Y \to X$ be part of a generator $(Y, i, \overline{d})$ for a $T_{\Sigma,E}$-algebra $\mathbb{X}$. Then any $x \in X$ admits some $\overline{d}(x) \in T_{\Sigma,E}Y$ such that $i^\sharp(\overline{d}(x)) = x$, where $i^\sharp : (T_{\Sigma,E}Y, \mu_Y) \to \mathbb{X}$. By construction $T_{\Sigma,E}Y = UFY$, where $FY = S_\Sigma Y/\cong_E$ is the set of Σ-terms generated by Y modulo the smallest congruence $\cong_E$ generated by E. Let $d(x) \in S_\Sigma Y$ be any representative of $\overline{d}(x) \in T_{\Sigma,E}Y$, that is, such that $[\![d(x)]\!]_E = \overline{d}(x)$. Then it follows $i^\sharp([\![d(x)]\!]_E) = i^\sharp(\overline{d}(x)) = x$.

Conversely, assume we have a $T_{\Sigma,E}$-algebra $\mathbb{X}$ and for any $x \in X$ there exists a term $d(x) \in S_\Sigma Y$ such that $i^\sharp([\![d(x)]\!]_E) = x$. Then we can define a function $\overline{d} : X \to T_{\Sigma,E}Y$ by $\overline{d}(x) = [\![d(x)]\!]_E$. It immediately follows $i^\sharp(\overline{d}(x)) = i^\sharp([\![d(x)]\!]_E) = x$, which shows that $(Y, i, \overline{d})$ is a generator for $\mathbb{X}$. □

5.3.7 Finitely Generated Objects

In this section, we relate our abstract definition of a generator to the theory of *locally finitely presentable* categories, in particular, to the notions of *finitely generated* and *finitely presentable* objects, which are categorical abstractions of finitely generated algebraic structures.

For intuition, recall that an element $x \in X$ of a partially ordered set is *compact*, if for each directed set $D \subseteq X$ with $x \leq \bigvee D$, there exists some $d \in D$ satisfying $x \leq d$. An *algebraic lattice* is a partially ordered set that has all joins, and every element is a join of compact elements. The naive categorification of compact elements is equivalent to the following definition: a object Y in $\mathscr{C}$ is *finitely presentable (generated)*, if $\mathsf{Hom}_{\mathscr{C}}(Y, -) : \mathscr{C} \to \mathsf{Set}$ preserves filtered colimits (of monomorphisms). Consequently, one can categorify algebraic lattices as *locally finitely presentable* (lfp) categories, which are cocomplete and admit a set of finitely presentable objects, such that every object is a filtered colimit of objects from that set [5].

In [7, Theor. 3.5] it is shown that an algebra $\mathbb{X}$ over a finitary monad T on an lfp category $\mathscr{C}$ is a finitely generated object of $\mathscr{C}^T$ iff there exists a finitely presentable object Y of $\mathscr{C}$ and a morphism $i : Y \to X$, such that $i^\sharp : (TY, \mu_Y) \to \mathbb{X}$ is a strong[5] epimorphism in $\mathscr{C}^T$. Below, we give a variant of this statement where instead the carrier of $i^\sharp$ is a split[6] epimorphism in $\mathscr{C}$, which is the case iff $\mathbb{X}$ admits a generator in the sense of Definition 4.4.0.1.

Proposition 5.3.7.1. *Let $\mathscr{C}$ be a lfp category in which strong epimorphisms split and T a finitary monad on $\mathscr{C}$ preserving epimorphisms. Then an algebra $\mathbb{X}$ over T is a finitely generated object of $\mathscr{C}^T$ iff it is generated by a finitely presentable object Y in $\mathscr{C}$ in the sense of Definition 4.4.0.1.*

Proof. Assume that an algebra $\mathbb{X}$ over T is a finitely generated object of $\mathscr{C}^T$. From [7, Theor. 3.5] it follows that there exists a finitely presentable object Y of $\mathscr{C}$ and a morphism $i : Y \to X$ such that $i^\sharp : (TY, \mu_Y) \to \mathbb{X}$ is a strong epimorphism in $\mathscr{C}^T$.

[5] An epimorphism $e : A \to B$ is said to be *strong*, if for any monomorphism $m : C \to D$ and any morphisms $f : A \to C$ and $g : B \to D$ such that $g \circ e = m \circ f$, there exists a diagonal monomorphism $d : B \to C$ such that $f = d \circ e$ and $g = m \circ d$.

[6] A morphism $e : A \to B$ is called *split*, if there exists a morphism $s : B \to A$ such that $e \circ s = \mathsf{id}_B$. Any morphism that is split is necessarily a strong epimorphism.

Since T preserves epis, it is sound to assume that the carrier $i^\sharp : TY \to X$ is a strong epimorphism in $\mathscr{C}$. (This is because the proof of [7, Theor. 3.5] can be modified by replacing the (strong-epi, mono)-factorisation system of the lfp category $\mathscr{C}^T$ (cf. [7, Remark 2.2.1] and [7, Remark 3.1]) with the factorisation system for $\mathscr{C}^T$ induced by lifting the (strong-epi, mono)-factorisation system of $\mathscr{C}$. The lifted factorisation system (cf. Section 5.2.2) consists of those algebra homomorphisms whose carrier is a strong-epi- or monomorphism in $\mathscr{C}$, respectively.) By assumption, $i^\sharp : TY \to X$ thus splits in $\mathscr{C}$, that is, there exists at least one morphism $d : X \to TY$ in $\mathscr{C}$ such that $i^\sharp \circ d = \mathsf{id}_X$. This shows that (Y, i, d) is a generator for $\mathbb{X}$ in the sense of Definition 4.4.0.1.

Conversely, assume that an algebra $\mathbb{X}$ over T is generated by (Y, i, d), where Y is a finitely presentable object in $\mathscr{C}$. Then d witnesses that $i^\sharp : TY \to X$ splits in $\mathscr{C}$. Since every split epimorphism is necessarily strong, $i^\sharp : TY \to X$ thus is a strong epimorphism in $\mathscr{C}$. It immediately follows that $i^\sharp : (TY, \mu_Y) \to \mathbb{X}$ is an epimorphism in $\mathscr{C}^T$. Since T preserves epis, it also is a *strong* epimorphism in $\mathscr{C}^T$. From [7, Theor. 3.5] it follows that the algebra $\mathbb{X}$ over T is a finitely generated object of $\mathscr{C}^T$. □

5.4 Related Work

One of the motivations for this chapter has been our broad interest in active learning algorithms for state-based models [14], in particular automata for NetKAT [12], a formal system for the verification of networks based on Kleene Algebra with Tests [92]. One of the main challenges in learning non-deterministic models such as NetKAT automata is the common lack of a unique minimal acceptor for a given language [49]. The problem has been independently approached for different variants of non-determinism, often with the common idea of finding a subclass admitting a unique representative [53, 28]. More general and unifying perspectives were given by van Heerdt [66, 63, 64] and Myers et al. [119]. One of the central notions in the work of van Heerdt is the concept of a scoop, originally introduced by Arbib and Manes [19].

In Chapter 4 we have presented a categorical framework that recovers minimal non-deterministic representatives in two steps. The framework is based on ideas closely related to the ones in [119], adopts scoops under the name generators and

strengthens the former to the notion of a basis (Definition 4.4.0.1). In a first step, the framework constructs the minimal bialgebra accepting a given regular language, by closing the minimal coalgebra with additional algebraic structure over a monad. In a second step, it identifies generators for the algebraic part of the bialgebra, to derive an equivalent coalgebra with side effects in a monad. In this chapter, we generalise the first step as application of a monad on an appropriate category of subobjects with respect to an $(\mathscr{E}, \mathscr{M})$-factorisation system, and explore the second step by further developing the abstract theory of generators and bases.

Categorical factorisation systems are well-established [36, 129, 106]. Among others, they have been used for a general view on the minimisation and determinisation of state-based systems [4, 6, 158]. In Section 5.2 we use the formalism of [4]. In Section 5.2.2 we have shown that under certain assumptions factorisation systems can be lifted to the categories of algebras and coalgebras. We later realised that the constructions had recently been published in [158].

The notion of a basis for an algebra over an arbitrary monad has been subject of previous interest. Jacobs, for instance, defines a basis as a coalgebra for the comonad on the category of algebras induced by the free algebra adjunction [77]. In Section 5.3.5 we have shown that a basis in our sense always induces a basis in their sense, and, conversely, it is possible to recover a basis in our sense from a basis in their sense, if certain assumptions about the existence and preservation of equaliser are given. As equaliser do not necessarily exist and are not necessarily preserved, our approach carries additional data and thus can be seen as finer.

5.5 Discussion and Future Work

We generalised the closure of a subset of an algebraic structure with respect to the latter as a monad between categories of subobjects relative to a factorisation system. We have identified the closure of a minimal coalgebra with additional algebraic structure as an instance of the closure of subobjects that arise by taking the image of a morphism. We have extended the notion of a generator to a category of algebras with generators, and explored its characteristics. We have generalised the matrix representation theory of vector spaces and discussed bases for bialgebras. We compared our

ideas with a coalgebraic generalisation of bases, explored the case in which a monad is induced by a variety, and briefly related our notion to finitely generated objects in finitely presentable categories.

In Chapter 4 we have shown that generators and bases in the sense of Section 5.3 are central ingredients in the definitions of minimal canonical acceptors. Many such acceptors admit double-reversal characterisations [38, 39, 119, 156]. Duality based characterisations as the former have been shown to be closely related to minimisation procedures with respect to factorisation systems [34, 33, 158]. In the future, it would be interesting to further explore the connection between the minimality of generators on the one side, and the notion of minimality of an acceptor with respect to a factorisation system on the other side.

Another interesting question is whether the construction that underlies our definition of a monad in Theorem 5.2.4.2 could be introduced at a more general level of an arbitrary adjunction between categories with suitable factorisation systems, such that the adjunction between the base category $\mathscr{C}$ and the category of Eilenberg-Moore algebras $\mathscr{C}^T$ is a special case.

Overall, our presentation primarily focused on applications to coalgebra. It would be valuable to also explore possible implications to other fields, for instance the minimisation of logical formulae or proofs.

Bibliography

[1] Fides Aarts and Frits Vaandrager. "Learning I/O Automata". In: *International Conference on Concurrency Theory*. Springer. 2010, pp. 71–85. DOI: 10.1007/978-3-642-15375-4_6.

[2] Fides Aarts, Falk Howar, Harco Kuppens, and Frits Vaandrager. "Algorithms for Inferring Register Automata". In: *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer. 2014, pp. 202–219. DOI: 10.1007/978-3-662-45234-9_15.

[3] Fides Aarts, Paul Fiterau-Brostean, Harco Kuppens, and Frits Vaandrager. "Learning Register Automata with Fresh Value Generation". In: *International Colloquium on Theoretical Aspects of Computing*. Springer. 2015, pp. 165–183. DOI: 10.1007/978-3-319-25150-9_11.

[4] Jiri Adamek, Horst Herrlich, and George E Strecker. "Abstract and Concrete Categories: The Joy of Cats". In: *Reprints in Theory and Applications of Categories* (2009).

[5] Jiri Adamek and Jiri Rosicky. *Locally Presentable and Accessible Categories*. Vol. 189. Cambridge University Press, 1994. DOI: 10.1017/CBO9780511600579.

[6] Jiri Adamek, Filippo Bonchi, Mathias Hülsbusch, Barbara König, Stefan Milius, and Alexandra Silva. "A Coalgebraic Perspective on Minimization and Determinization". In: *International Conference on Foundations of Software Science and Computational Structures*. Springer. 2012, pp. 58–73. DOI: 10.1007/978-3-642-28729-9_4.

[7] Jiri Adamek, Stefan Milius, Lurdes Sousa, and Thorsten Wißmann. "Finitely Presentable Algebras For Finitary Monads". In: *Theory and Applications of Categories* 34.37 (2019), pp. 1179–1195.

[8] Jürgen Albert and Jarkko Kari. "Digital Image Compression". In: *Handbook of Weighted Automata*. Springer, 2009, pp. 453–479. DOI: 10.1007/978-3-642-01492-5_11.

[9] Cyril Allauzen, Mehryar Mohri, and Ameet Talwalkar. "Sequence Kernels for Predicting Protein Essentiality". In: *Proceedings of the 25th International Conference on Machine Learning*. 2008, pp. 9–16. DOI: 10.1145/1390156.1390158.

[10] Bowen Alpern and Fred B Schneider. "Recognizing Safety and Liveness". In: *Distributed Computing* 2.3 (1987), pp. 117–126. DOI: 10.1007/BF01782772.

[11] Benjamin Aminof, Orna Kupferman, and Robby Lampert. "Formal Analysis of Online Algorithms". In: *International Symposium on Automated Technology for Verification and Analysis*. Springer. 2011, pp. 213–227. DOI: 10.1007/978-3-642-24372-1_16.

[12] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. "NetKAT: Semantic Foundations for Networks". In: *ACM SIGPLAN Notices* 49.1 (2014), pp. 113–126. DOI: 10.1145/2578855.2535862.

[13] Dana Angluin. "A Note on the Number of Queries Needed to Identify Regular Languages". In: *Information and Control* 51.1 (1981), pp. 76–87. DOI: 10.1016/S0019-9958(81)90090-5.

[14] Dana Angluin. "Learning Regular Sets from Queries and Counterexamples". In: *Information and Computation* 75.2 (1987), pp. 87–106. DOI: 10.1016/0890-5401(87)90052-6.

[15] Dana Angluin. "Queries and Concept Learning". In: *Machine Learning* 2.4 (1988), pp. 319–342. DOI: 10.1023/A:1022821128753.

[16] Dana Angluin and Miklós Csundefinedrös. "Learning Markov Chains with Variable Memory Length from Noisy Output". In: *Proceedings of the Tenth Annual Conference on Computational Learning Theory*. Association for Computing Machinery, 1997, pp. 298–308. DOI: 10.1145/267460.267517.

[17] Dana Angluin, Sarah Eisenstat, and Dana Fisman. "Learning Regular Languages via Alternating Automata". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. 2015, pp. 3308–3314.

[18] Dana Angluin and Dana Fisman. "Learning Regular Omega Languages". In: *Theoretical Computer Science* 650 (2016), pp. 57–72. DOI: 10.1016/j.tcs.2016.07.031.

[19] Michael A Arbib and Ernest G Manes. "Fuzzy Machines in a Category". In: *Bulletin of the Australian Mathematical Society* 13.2 (1975), pp. 169–210. DOI: 10.1017/S0004972700024412.

[20] André Arnold, Anne Dicky, and Maurice Nivat. "A Note About Minimal Non-Deterministic Automata". In: *Bulletin of the EATCS* 47 (1992), pp. 166–169.

[21] Steve Awodey. *Category Theory*. Oxford University Press, Inc., 2010.

[22] Borja Balle and Mehryar Mohri. "Learning Weighted Automata". In: *International Conference on Algebraic Informatics*. Springer. 2015, pp. 1–21. DOI: 10.1007/978-3-319-23021-4_1.

[23] Borja Balle and Mehryar Mohri. "Spectral Learning of General Weighted Automata via Constrained Matrix Completion". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[24] Alexandru Baltag. "A Logic for Coalgebraic Simulation". In: *Electronic Notes in Theoretical Computer Science* 33 (2000), pp. 42–60. DOI: 10.1016/S1571-0661(05)80343-3.

[25] Jon Beck. "Distributive Laws". In: *Seminar on Triples and Categorical Homology Theory*. Springer. 1969, pp. 119–140. DOI: 10.1007/BFb0083084.

[26] Francesco Bergadano and Stefano Varricchio. "Learning Behaviors of Automata from Multiplicity and Equivalence Queries". In: *Algorithms and Complexity*. Springer, 1994, pp. 54–62. DOI: 10.1007/3-540-57811-0_6.

[27] Francesco Bergadano and Stefano Varricchio. "Learning Behaviors of Automata from Multiplicity and Equivalence Queries". In: *SIAM Journal on Computing* 25.6 (1996), pp. 1268–1280. DOI: 10.1137/S009753979326091X.

[28] Sebastian Berndt, Maciej Liśkiewicz, Matthias Lutter, and Rüdiger Reischuk. "Learning Residual Alternating Automata". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017. DOI: 10.1609/aaai.v31i1.10891.

[29] Garrett Birkhoff. "On the Structure of Abstract Algebras". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 31. 4. Cambridge University Press. 1935, pp. 433–454.

[30] Benedikt Bollig, Peter Habermehl, Martin Leucker, and Benjamin Monmege. "A Fresh Approach to Learning Register Automata". In: *International Conference on Developments in Language Theory*. Springer. 2013, pp. 118–130. DOI: 10.1007/978-3-642-38771-5_12.

[31] Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. "Angluin-Style Learning of NFA". In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. IJCAI'09. 2009, 1004–1009.

[32] Filippo Bonchi and Damien Pous. "Checking NFA Equivalence With Bisimulations up to Congruence". In: *ACM SIGPLAN Notices* 48.1 (2013), pp. 457–468.

[33] Filippo Bonchi, Marcello M Bonsangue, Helle H Hansen, Prakash Panangaden, Jan Rutten, and Alexandra Silva. "Algebra-Coalgebra Duality in Brzozowski's Minimization Algorithm". In: *ACM Transactions on Computational Logic (TOCL)* 15.1 (2014), pp. 1–29. DOI: 10.1145/2490818.

[34] Filippo Bonchi, Marcello M Bonsangue, Jan Rutten, and Alexandra Silva. "Brzozowski's Algorithm (Co)Algebraically". In: *Logic and Program Semantics*. Springer, 2012, pp. 12–23. DOI: 10.1007/978-3-642-29485-3_2.

[35] Marcello M Bonsangue, Helle Hvid Hansen, Alexander Kurz, and Jurriaan Rot. "Presenting Distributive Laws". In: *International Conference on Algebra and Coalgebra in Computer Science*. Springer. 2013, pp. 95–109. DOI: 10.1007/978-3-642-40206-7_9.

[36] Aldridge K Bousfield. "Constructions of Factorization Systems in Categories". In: *Journal of Pure and Applied Algebra* 9.2-3 (1977), pp. 207–220. DOI: 10.1016/0022-4049(77)90067-6.

[37] Thomas M Breuel. "The OCRopus open source OCR system". In: *Document Recognition and Retrieval XV*. Vol. 6815. International Society for Optics and Photonics. SPIE, 2008, pp. 120–134. DOI: 10.1117/12.783598.

[38] Janusz A Brzozowski. "Canonical Regular Expressions and Minimal State Graphs for Definite Events". In: *Proc. Symposium of Mathematical Theory of Automata*. Vol. 12. 1962, pp. 529–561.

[39] Janusz A. Brzozowski and Hellis Tamm. "Theory of Átomata". In: *Theor. Comput. Sci.* 539 (2014), pp. 13–27. DOI: 10.1016/j.tcs.2014.04.016.

[40] Hugues Calbrix, Maurice Nivat, and Andreas Podelski. "Ultimately Periodic Words of Rational ω-Languages". In: *International Conference on Mathematical Foundations of Programming Semantics*. Springer. 1993, pp. 554–566. DOI: 10.1007/3-540-58027-1_27.

[41] Sofia Cassel, Falk Howar, Bengt Jonsson, and Bernhard Steffen. "Active Learning for Extended Finite State Machines". In: *Formal Aspects of Computing* 28.2 (2016), pp. 233–263. DOI: 10.1007/s00165-016-0355-5.

[42] Georg Chalupar, Stefan Peherstorfer, Erik Poll, and Joeri De Ruiter. "Automated Reverse Engineering using Lego®". In: *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. USENIX Association, 2014.

[43] T.S. Chow. "Testing Software Design Modeled by Finite-State Machines". In: *IEEE Transactions on Software Engineering* SE-4.3 (1978), pp. 178–187. DOI: 10.1109/TSE.1978.231496.

[44] Dion Coumans and Bart Jacobs. "Scalars, Monads, and Categories". In: *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*. Oxford University Press, 2013. DOI: 10.1093/acprof:oso/9780199646296.003.0007.

[45] Karel Culik II and Jarkko Kari. "Image Compression Using Weighted Finite Automata". In: *Computers & Graphics* 17.3 (1993), pp. 305–313. DOI: 10.1016/0097-8493(93)90079-O.

[46] Fredrik Dahlqvist and Todd Schmid. "How to Write a Coequation". In: *9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021)*. Leibniz International Proceedings in Informatics (LIPIcs). 2021, 13:1–13:25. DOI: 10.4230/LIPIcs.CALCO.2021.13.

[47] Loris D'Antoni and Margus Veanes. "Minimization of Symbolic Automata". In: *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '14. Association for Computing Machinery, 2014, pp. 541–553. DOI: 10.1145/2535838.2535849.

[48] Joeri De Ruiter and Erik Poll. "Protocol State Fuzzing of TLS Implementations". In: *Proceedings of the 24th USENIX Conference on Security Symposium*. SEC'15. USENIX Association, 2015, pp. 193–206.

[49] François Denis, Aurélien Lemay, and Alain Terlutte. "Residual Finite State Automata". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 2001, pp. 144–157. DOI: 10.1007/3-540-44693-1_13.

[50] Samuel Drews and Loris D'Antoni. "Learning Symbolic Automata". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2017, pp. 173–189. DOI: 10.1007/978-3-662-54577-5_10.

[51] Loris D'Antoni, Tiago Ferreira, Matteo Sammartino, and Alexandra Silva. "Symbolic Register Automata". In: *International Conference on Computer Aided Verification*. Springer. 2019, pp. 3–21. DOI: 10.1007/978-3-030-25540-4_1.

[52] Samuel Eilenberg, John C. Moore, et al. "Adjoint Functors and Triples". In: *Illinois Journal of Mathematics* 9.3 (1965), pp. 381–398. DOI: 10.1215/ijm/1256068141.

[53] Yann Esposito, Aurélien Lemay, François Denis, and Pierre Dupont. "Learning Probabilistic Residual Finite State Automata". In: *International Colloquium on Grammatical Inference*. Springer. 2002, pp. 77–91. DOI: 10.1007/3-540-45790-9_7.

[54] Azadeh Farzan, Yu Fang Chen, Edmund M. Clarke, Yih Kuen Tsay, and Bow Yaw Wang. "Extending Automated Compositional Verification to the Full Class of Omega-Regular Languages". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 2–17.

[55] Nick Feamster, Jennifer Rexford, and Ellen Zegura. "The Road to SDN: An Intellectual History of Programmable Networks". In: *ACM SIGCOMM Computer Communication Review* 44.2 (2014), pp. 87–98. DOI: 10.1145/2602204.2602219.

[56] Dana Fisman, Hadar Frenkel, and Sandra Zilles. "Inferring Symbolic Automata". In: *30th EACSL Annual Conference on Computer Science Logic (CSL 2022)*. Vol. 216. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 21:1–21:19. DOI: 10.4230/LIPIcs.CSL.2022.21.

[57] Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. "A Coalgebraic Decision Procedure for NetKAT". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '15. Association for Computing Machinery, 2015, pp. 343–355. DOI: 10.1145/2676726.2677011.

[58] Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. "Probabilistic NetKAT". In: *Proceedings of the 25th European Symposium on Programming Languages and Systems*. Springer, 2016, pp. 282–309. DOI: 10.1007/978-3-662-49498-1_12.

[59] Georgios Giantamidis and Stavros Tripakis. "Learning Moore Machines from Input-Output Traces". In: *FM 2016: Formal Methods*. Springer, 2016, pp. 291–309. DOI: 10.1007/978-3-319-48989-6_18.

[60] Niels Bjørn Bugge Grathwohl, Dexter Kozen, and Konstantinos Mamouras. "KAT + B!" In: *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science*. CSL-LICS '14. Association for Computing Machinery, 2014, pp. 1–10. DOI: 10.1145/2603088.2603095.

[61] Andreas Hagerer, Hardi Hungar, Oliver Niese, and Bernhard Steffen. "Model Generation by Moderated Regular Extrapolation". In: *International Conference on Fundamental Approaches to Software Engineering*. Springer. 2002, pp. 80–95. DOI: 10.1007/3-540-45923-5_6.

[62] Helle Hvid Hansen, Clemens Kupke, and Raul Andres Leal. "Strong Completeness for Iteration-Free Coalgebraic Dynamic Logics". In: *IFIP International Conference on Theoretical Computer Science*. Springer. 2014, pp. 281–295. DOI: 10.1007/978-3-662-44602-7_22.

[63] Gerco van Heerdt. "An Abstract Automata Learning Framework". MA thesis. Radboud University Nijmegen, 2016.

[64] Gerco van Heerdt. "CALF: Categorical Automata Learning Framework". PhD thesis. University College London, 2020.

[65] Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. "CALF: Categorical Automata Learning Framework". In: *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*. Vol. 82. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 29:1–29:24. DOI: 10.4230/LIPIcs.CSL.2017.29.

[66] Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. "Learning Automata with Side-Effects". In: *Coalgebraic Methods in Computer Science*. Springer, 2020, pp. 68–89. DOI: 10.1007/978-3-030-57201-3_5.

[67] Gerco van Heerdt, Joshua Moerman, Matteo Sammartino, and Alexandra Silva. "A (Co)Algebraic Theory of Succinct Automata". In: *Journal of Logical and Algebraic Methods in Programming* 105 (2019), pp. 112–125. DOI: 10.1016/j.jlamp.2019.02.008.

[68] Gerco van Heerdt, Clemens Kupke, Jurriaan Rot, and Alexandra Silva. "Learning Weighted Automata over Principal Ideal Domains". In: *Foundations of Software Science and Computation Structures*. Springer, 2020, pp. 602–621. DOI: 10.1007/978-3-030-45231-5_31.

[69] Wim H. Hesselink and Albert Thijs. "Fixpoint Semantics and Simulation". In: *Theoretical Computer Science* 238.1 (2000), pp. 275–311. DOI: 10.1016/S0304-3975(98)00176-5.

[70] John E Hopcroft and Richard M Karp. *A Linear Algorithm for Testing Equivalence of Finite Automata*. Vol. 114. Defense Technical Information Center, 1971.

[71] Falk Howar. "Active Learning of Interface Programs". PhD thesis. Dortmund University of Technology, 2012. DOI: 10.17877/DE290R-4817.

[72] Falk Howar, Bernhard Steffen, Bengt Jonsson, and Sofia Cassel. "Inferring Canonical Register Automata". In: *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer. 2012, pp. 251–266.

[73] Malte Isberner, Falk Howar, and Bernhard Steffen. "Learning Register Automata: From Languages to Program Structures". In: *Machine Learning* 96.1 (2014), pp. 65–98. DOI: 10.1007/s10994-013-5419.

[74] Malte Isberner, Falk Howar, and Bernhard Steffen. "The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning". In: *International Conference on Runtime Verification*. Springer. 2014, pp. 307–322. DOI: 10.1007/978-3-319-11164-3_26.

[75] Bart Jacobs. "A Bialgebraic Review of Deterministic Automata, Regular Expressions and Languages". In: *Algebra, Meaning, and Computation*. Springer, 2006, pp. 375–404. DOI: 10.1007/11780274_20.

[76] Bart Jacobs. "A Recipe for State-and-Effect Triangles". In: *6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2015.

[77] Bart Jacobs. "Bases as Coalgebras". In: *Algebra and Coalgebra in Computer Science*. Springer. 2011, pp. 237–252. DOI: 10.1007/978-3-642-22944-2_17.

[78] Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016. DOI: 10.1017/CBO9781316823187.

[79] Bart Jacobs and Jesse Hughes. "Simulations in Coalgebra". In: *Electronic Notes in Theoretical Computer Science* 82.1 (2003), pp. 128–149. ISSN: 1571-0661. DOI: 10.1016/S1571-0661(04)80636-4.

[80] Bart Jacobs and Alexandra Silva. "Automata Learning: A Categorical Perspective". In: *Horizons of the Mind. A Tribute to Prakash Panangaden*. Springer, 2014, pp. 384–406. DOI: 10.1007/978-3-319-06880-0_20.

[81] Bart Jacobs, Alexandra Silva, and Ana Sokolova. "Trace Semantics via Determinization". In: *International Workshop on Coalgebraic Methods in Computer Science*. Springer. 2012, pp. 109–129. DOI: 10.1007/978-3-642-32784-1_7.

[82] Michael Kaminski and Nissim Francez. "Finite-Memory Automata". In: *Theoretical Computer Science* 134.2 (1994), pp. 329–363. DOI: 10.1016/0304-3975(94)90242-9.

[83] Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. DOI: 10.7551/mitpress/3897.001.0001.

[84] S. C. Kleene. "Representation of Events in Nerve Nets and Finite Automata". In: *Automata Studies*. Vol. 34. Princeton University Press, 1956, pp. 3–42. DOI: 10.1515/9781400882618-002.

[85] Bartek Klin. "A Coalgebraic Approach to Process Equivalence and a Coinduction Principle for Traces". In: *Electronic Notes in Theoretical Computer Science* 106 (2004), pp. 201–218. DOI: 10.1016/j.entcs.2004.02.029.

[86] Bartek Klin. "Bialgebras for Structural Operational Semantics: An Introduction". In: *Theoretical Computer Science* 412.38 (2011), pp. 5043–5069. DOI: 10.1016/j.tcs.2011.03.023.

[87] Bartek Klin and Beata Nachyla. "Presenting Morphisms of Distributive Laws". In: *6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015)*. Vol. 35. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2015, 190–204. DOI: 10.4230/LIPIcs.CALCO.2015.190.

[88] Dexter Kozen. "A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events". In: *Information and Computation* 110.2 (1994), pp. 366–390. DOI: 10.1006/inco.1994.1037.

[89] Dexter Kozen. *Automata on Guarded Strings and Applications*. Tech. rep. Cornell University, 2001.

[90] Dexter Kozen. "Kleene Algebra with Tests". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 19.3 (1997), pp. 427–443. DOI: 10.1145/256167.256195.

[91] Dexter Kozen. "On the Coalgebraic Theory of Kleene Algebra with Tests". In: *Rohit Parikh on Logic, Language and Society*. Springer, 2017, pp. 279–298. DOI: 10.1007/978-3-319-47843-2_15.

[92] Dexter Kozen and Frederick Smith. "Kleene Algebra with Tests: Completeness and Decidability". In: *International Workshop on Computer Science Logic*. Springer. 1997, pp. 244–259. DOI: 10.1007/3-540-63172-0_43.

[93] Dexter Kozen and Wei-Lung Dustin Tseng. "The Böhm–Jacopini Theorem is False, Propositionally". In: *International Conference on Mathematics of Program Construction*. Springer, 2008, pp. 177–192. DOI: 10.1007/978-3-540-70594-9_11.

[94] Dexter C. Kozen. *Automata and Computability*. 1st. Springer, 1997. DOI: 10.1007/978-1-4612-1844-9.

[95] Alexander Kurz. "Logics for Coalgebras and Applications to Computer Science". PhD thesis. Ludwig-Maximilians-Universität München, 2000.

[96] Serge Lang. "Algebra". In: *Graduate Texts in Mathematics* (2002).

[97] Sławomir Lasota, Bartek Klin, and Mikołaj Bojańczyk. "Automata Theory in Nominal Sets". In: *Logical Methods in Computer Science* 10.3 (2014). DOI: 10.2168/LMCS-10(3:4)2014.

[98] Chin Soon Lee, Neil D Jones, and Amir M Ben-Amram. "The Size-Change Principle for Program Termination". In: *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '01. Association for Computing Machinery, 2001, pp. 81–92. DOI: 10.1145/360204.360210.

[99] Tom Leinster. *Basic Category Theory*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107360068.

[100] Marina Lenisa, John Power, and Hiroshi Watanabe. "Distributivity for Endofunctors, Pointed and Co-pointed Endofunctors, Monads and Comonads". In: *Electronic Notes in Theoretical Computer Science* 33 (2000), pp. 230–260. DOI: 10.1016/S1571-0661(05)80350-0.

[101] Paul Blain Levy. "Similarity Quotients as Final Coalgebras". In: *Foundations of Software Science and Computational Structures*. Springer. 2011, pp. 27–41. DOI: 10.1007/978-3-642-19805-2_3.

[102] Yong Li, Yu-Fang Chen, Lijun Zhang, and Depeng Liu. "A Novel Learning Algorithm for Büchi Automata Based on Family of DFAs and Classification Trees". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Vol. 281. Springer, 2017, pp. 208–226. DOI: 10.1007/978-3-662-54577-5_12.

[103] Yong Li, Yu-Fang Chen, Lijun Zhang, and Depeng Liu. "A Novel Learning Algorithm for Büchi Automata Based on Family of DFAs and Classification Trees". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2017, pp. 208–226.

[104] Fred EJ Linton. "Some Aspects of Equational Categories". In: *Proceedings of the Conference on Categorical Algebra*. Springer. 1966, pp. 84–94. DOI: 10.1007/978-3-642-99902-4_3.

[105] Saunders Mac Lane. *Categories for the Working Mathematician*. Vol. 5. Springer, 2013. DOI: 10.1007/978-1-4757-4721-8.

[106] Saunders MacLane. "Duality for Groups". In: *Bulletin of the American Mathematical Society* 56.6 (1950), pp. 485–516.

[107] Oded Maler and Irini-Eleftheria Mens. "Learning Regular Languages over Large Alphabets". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2014, pp. 485–499. DOI: 10.1007/978-3-642-54862-8_41.

[108] Oded Maler and Amir Pnueli. "On the Learnability of Infinitary Regular Sets". In: *Information and Computation* 118.2 (1995), pp. 316–326. DOI: 10.1006/inco.1995.1070.

[109] George H. Mealy. "A Method for Synthesizing Sequential Circuits". In: *The Bell System Technical Journal* 34.5 (1955), pp. 1045–1079. DOI: 10.1002/j.1538-7305.1955.tb03788.x.

[110] Joshua Moerman. "Learning Product Automata". In: *Proceedings of The 14th International Conference on Grammatical Inference 2018*. Vol. 93. Proceedings of Machine Learning Research. PMLR, 2019, pp. 54–66.

[111] Joshua Moerman and Matteo Sammartino. "Residual Nominal Automata". In: *CONCUR*. Vol. 171. 2020, 44:1–44:21. DOI: 10.4230/LIPIcs.CONCUR.2020.44.

[112] Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michał Szynwelski. "Learning Nominal Automata". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. POPL '17. Association for Computing Machinery, 2017, 613–625. DOI: 10.1145/3009837.3009879.

[113] Eugenio Moggi. *An Abstract View of Programming Languages*. University of Edinburgh, Department of Computer Science, Laboratory for Foundations of Computer Science, 1990.

[114] Eugenio Moggi. *Computational Lambda-Calculus and Monads*. University of Edinburgh, Department of Computer Science, Laboratory for Foundations of Computer Science, 1988.

[115] Eugenio Moggi. "Notions of Computation and Monads". In: *Information and Computation* 93.1 (1991), pp. 55–92. DOI: 10.1016/0890-5401(91)90052-4.

[116] Mehryar Mohri. "Weighted Automata Algorithms". In: *Handbook of Weighted Automata*. Springer, 2009, pp. 213–254. DOI: 10.1007/978-3-642-01492-5_6.

[117] Mehryar Mohri, Fernando Pereira, and Michael Riley. "Speech Recognition with Weighted Finite-State Transducers". In: *Springer Handbook of Speech Processing*. Springer, 2008, pp. 559–584. DOI: 10.1007/978-3-540-49127-9_28.

[118] Tyler Moore. "Gedanken–experiments on Sequential Machines". In: *Automata Studies, Annals of Mathematical Studies, no. 34*. Citeseer. 1956.

[119] Robert S. R. Myers, Jiri Adamek, Stefan Milius, and Henning Urbat. "Coalgebraic Constructions of Canonical Nondeterministic Automata". In: *Theoretical Computer Science* 604 (2015), pp. 81–101. DOI: 10.1016/j.tcs.2015.03.035.

[120] Anil Nerode. "Linear Automaton Transformations". In: *Proceedings of the American Mathematical Society* 9.4 (1958), pp. 541–544. DOI: 10.2307/2033204.

[121] *Number of join-irreducible elements of a lattice: is it monotonic?* URL: https://math.stackexchange.com/questions/801833/number-of-join-irreducible-elements-of-a-lattice-is-it-monotonic (visited on 03/29/2021).

[122] *OCaml*. URL: https://ocaml.org.

[123] Louis Parlant, Jurriaan Rot, Alexandra Silva, and Bas Westerbaan. "Preservation of Equations by Monoidal Monads". In: *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*. Vol. 170. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020, 77:1–77:14. DOI: 10.4230/LIPIcs.MFCS.2020.77.

[124] Jorge M Pena and Arlindo L Oliveira. "A New Algorithm for the Reduction of Incompletely Specified Finite State Machines". In: *1998 IEEE/ACM International Conference on Computer-Aided Design*. 1998, pp. 482–489. DOI: 10.1145/288548.289075.

[125] Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Vol. 57. Cambridge University Press, 2013.

[126] Amir Pnueli and Roni Rosner. "On the Synthesis of a Reactive Module". In: *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '89. Association for Computing Machinery, 1989, pp. 179–190. DOI: 10.1145/75277.75293.

[127] Damien Pous. "Symbolic Algorithms for Language Equivalence and Kleene Algebra with Tests". In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '15. Association for Computing Machinery, 2015, pp. 357–368. DOI: 10.1145/2676726.2677007.

[128] John Power and Hiroshi Watanabe. "Combining a Monad and a Comonad". In: *Theoretical Computer Science* 280.1 (2002), pp. 137–162. DOI: https://doi.org/10.1016/S0304-3975(01)00024-X.

[129] Emily Riehl. "Factorization Systems". In: (2008). URL: https://math.jhu.edu/~eriehl/factorization.pdf (visited on 11/28/2022).

[130] Ronald L Rivest and Robert E Schapire. "Inference of Finite Automata Using Homing Sequences". In: *Information and Computation* 103.2 (1993), pp. 299–347. DOI: 10.1006/inco.1993.1021.

[131] Jan Rutten. "The Method of Coalgebra: Exercises in Coinduction". In: (2019).

[132] Jan Rutten. "Universal Coalgebra: A Theory of Systems". In: *Theoretical Computer Science* 249.1 (2000), pp. 3–80. DOI: 10.1016/S0304-3975(00)00056-6.

[133] Jan Rutten, Marcello Bonsangue, Filippo Bonchi, and Alexandra Silva. "Generalizing Determinization from Automata to Coalgebras". In: *Logical Methods in Computer Science* 9.1 (2013). DOI: 10.2168/LMCS-9(1:9)2013.

[134] Arto Salomaa and Matti Soittola. "Automata-Theoretic Aspects of Formal Power Series". In: *Texts and Monographs in Computer Science*. Springer, 1978. DOI: 10.1007/978-1-4612-6264-0.

[135] T Schmid, T Kappé, D Kozen, and A Silva. "Guarded Kleene Algebra with Tests: Coequations, Coinduction, and Completeness". In: *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Vol. 198. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, p. 142. DOI: 10.4230/LIPIcs.ICALP.2021.142.

[136] Lutz Schröder. "Expressivity of Coalgebraic Modal Logic: The Limits and Beyond". In: *Theoretical Computer Science* 390.2 (2008), pp. 230–247. DOI: 10.1016/j.tcs.2007.09.023.

[137] Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. "Nominal Automata with Name Binding". In: *Foundations of Software Science and Computation Structures*. Springer, 2017, pp. 124–142. DOI: 10.1007/978-3-662-54458-7_8.

[138] Gavin J Seal. "Tensors, Monads and Actions". In: *Theory and Applications of Categories* 28.15 (2013), pp. 403–433.

[139] Muzammil Shahbaz and Roland Groz. "Inferring Mealy Machines". In: *International Symposium on Formal Methods*. Springer. 2009, pp. 207–222. DOI: 10.1007/978-3-642-05089-3_14.

[140] Alexandra Silva, Filippo Bonchi, Marcello M Bonsangue, and Jan Rutten. "Generalizing the Powerset Construction, Coalgebraically". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*. Vol. 8. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2010, pp. 272–283. DOI: 10.4230/LIPIcs.FSTTCS.2010.272.

[141] Steffen Smolka, Spiridon Eliopoulos, Nate Foster, and Arjun Guha. "A Fast Compiler for NetKAT". In: *Proceedings of the 20th ACM SIGPLAN International Conference on Functional Programming*. 2015, pp. 328–341.

[142] Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. "Guarded Kleene Algebra with Tests: Verification of Uninterpreted Programs in Nearly Linear Time". In: *Proceedings of the ACM on Programming Languages* 4.POPL (2019), pp. 1–28. DOI: 10.1145/3371129.

[143] Steffen Smolka, Praveen Kumar, David M Kahn, Nate Foster, Justin Hsu, Dexter Kozen, and Alexandra Silva. "Scalable Verification of Probabilistic Networks". In: *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI 2019. Association for Computing Machinery, 2019, pp. 190–203. DOI: 10.1145/3314221.3314639.

[144] Bernhard Steffen, Falk Howar, and Malte Isberner. "Active Automata Learning: From DFAs to Interface Programs and Beyond". In: *Proceedings of the Eleventh International Conference on Grammatical Inference*. Vol. 21. Proceedings of Machine Learning Research. PMLR, 2012, pp. 195–209.

[145] Bernhard Steffen, Falk Howar, and Maik Merten. "Introduction to Active Automata Learning From a Practical Perspective". In: *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. Springer. 2011, pp. 256–296.

[146] Ross Street. "The Formal Theory of Monads". In: *Journal of Pure and Applied Algebra* 2.2 (1972), pp. 149–168. DOI: 10.1016/0022-4049(72)90019-9.

[147] Ross Street. "Weak Distributive Laws". In: *Theory and Applications of Categories* 22 (2009), pp. 313–320.

[148] Hellis Tamm and Margus Veanes. "Theoretical Aspects of Symbolic Automata". In: *SOFSEM 2018: Theory and Practice of Computer Science*. Springer, 2018, pp. 428–441. DOI: 10.1007/978-3-319-73117-9_30.

[149] Paul Taylor. "Subspaces in Abstract Stone Duality". In: *Theory and Applications of Categories* 10.13 (2002), pp. 301–368.

[150] Ken Thompson. "Programming Techniques: Regular Expression Search Algorithm". In: *Communications of the ACM* 11.6 (1968), pp. 419–422. DOI: 10.1145/363347.363387.

[151] Daniele Turi. "Functorial Operational Semantics". PhD thesis. Vrije Universiteit Amsterdam, 1996.

[152] Daniele Turi and Gordon Plotkin. "Towards a Mathematical Operational Semantics". In: *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1997, pp. 280–291. DOI: 10.1109/LICS.1997.614955.

[153] Frits Vaandrager. "Model Learning". In: *Communications of the ACM* 60.2 (2017), pp. 86–95. DOI: 10.1145/2967606.

[154] Frits Vaandrager, Bharat Garhewal, Jurriaan Rot, and Thorsten Wißmann. "A New Approach for Active Automata Learning Based on Apartness". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2022, pp. 223–243. DOI: 10.1007/978-3-030-99524-9_12.

[155] Moshe Y Vardi and Pierre Wolper. "An Automata-Theoretic Approach to Automatic Program Verification". In: *Proceedings of the Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 1986, pp. 332–344.

[156] Jean Vuillemin and Nicolas Gama. *Efficient Equivalence and Minimization for Non Deterministic Xor Automata*. Tech. rep. Ecole Normale Supérieure, 2010.

[157] Hiroshi Watanabe. "Well-Behaved Translations Between Structural Operational Semantics". In: *Electronic Notes in Theoretical Computer Science* 65.1 (2002), pp. 337–357. DOI: 10.1016/S1571-0661(04)80372-4.

[158] Thorsten Wißmann. "Minimality Notions via Factorization Systems and Examples". In: *Logical Methods in Computer Science* 18.3 (2022). DOI: 10.46298/lmcs-18(3:31)2022.

[159] Stefan Zetzsche, Alexandra Silva, and Matteo Sammartino. "Generators and Bases for Monadic Closures". In: *arXiv preprint arXiv:2010.10223* (2023).

[160] Stefan Zetzsche, Alexandra Silva, and Matteo Sammartino. "Guarded Kleene Algebra with Tests: Automata Learning". In: *Electronic Notes in Theoretical Informatics and Computer Science* Volume 1 - Proceedings of MFPS XXXVIII (2023). DOI: 10.46298/entics.10505.

[161] Stefan Zetzsche, Gerco van Heerdt, Matteo Sammartino, and Alexandra Silva. "Canonical Automata via Distributive Law Homomorphisms". In: *Electronic Proceedings in Theoretical Computer Science* 351 (2021), 296–313. DOI: 10.4204/eptcs.351.18.

[162] Maaike Zwart and Dan Marsden. "No-Go Theorems for Distributive Laws". In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 2019, pp. 1–13. DOI: 10.1109/LICS.2019.8785707.

List of Figures