EXTENSIONS OF KLEENE ALGEBRA FOR PROGRAM VERIFICATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Konstantinos Mamouras

August 2015

EXTENSIONS OF KLEENE ALGEBRA FOR PROGRAM VERIFICATION

Konstantinos Mamouras, Ph.D.

Cornell University 2015

Kleene algebra (KA) is an algebraic system that captures completely the laws of equivalence for regular expressions. It is also useful for reasoning about a multitude of computationally interesting structures. Of central interest in KA is a "star" operation that typically describes some kind of repetition or *iteration*. A combination of Kleene algebra with Boolean algebra has been proposed under the name of *Kleene algebra with tests* (KAT). This logical system can model the conventional programming constructs of imperative iterative programs, e.g. while loops and conditionals, and it has proven useful for various program verification tasks.

In this dissertation we investigate variations and extensions of KA and KAT that are useful for program verification and for reasoning about mathematical structures that appear in computer science. At a technical level, we are interested in establishing *completeness theorems*, which assert that a proposed logical system is strong enough to capture all properties that are true in a mathematical structure or in a class of mathematical structures. Such results assure us of the quality of a logical system, and their proof often (certainly for the systems that we study here) reveals a systematic way of creating *proof objects* that certify properties of interest.

The first part of this thesis explores generic extensions of KA and KAT with *extra equational assumptions*. Such extensions are meant to capture in the context of program verification some crucial properties of the domain of computation.

We present a very general completeness meta-theorem that instantiates into several useful concrete completeness results.

The second part focuses on extensions of KAT with *extra mutable state* that enable many useful semantics-preserving program transformations. The transformations we intend to cover are motivated by classical examples from the area of program schematology. We offer a rigorous and mathematically appealing algebraic approach, which replaces the typical combinatorial arguments about flowchart schemes with equational reasoning.

Finally, a typed variation of KA is investigated that intends to capture properties of mathematical structures that appear in domain theory. The main result is that our proposed *typed KA with products* is strong enough to establish all the abstract properties of parametric fixpoints in the standard semantic model $\mathbf{CPO}$, which is the category of ω-CPOs and ω-continuous maps.

BIOGRAPHICAL SKETCH

Konstantinos Mamouras was born in Athens, Greece. He received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens in April 2008, and a MSc in Advanced Computing from Imperial College London in September 2009. He continued with graduate studies at Cornell University, where he obtained a MSc in Computer Science in May 2014 and expects to receive a PhD in Computer Science in August 2015.

To my parents, Ioannis and Stavroula.

ACKNOWLEDGEMENTS

I am extremely fortunate to have spent my years as a PhD student at Cornell University. First of all, I want to thank my advisor, Dexter Kozen, without whom this thesis would not have been written. I am extremely grateful for the direction and insights he provided to my research endeavor, and for teaching me more than I could have hoped to learn during my PhD studies. I also want to thank my other two committee members, Robert Constable and Richard Shore, for their invaluable comments and suggestions.

For one semester of my studies I had the pleasure of being hosted by Alexandra Silva at Radboud University Nijmegen. I am very thankful to Alexandra for the wonderful collaboration envinroment that I enjoyed there. I will surely remember that semester as one of the most enjoyable of my PhD years.

Finally, I want to thank my family and friends. Their encouragement and support made the five-year long pursuit of a doctorate much more enjoyable.

TABLE OF CONTENTS

Biographical Sketch iii
Dedication iv
Acknowledgements v
Table of Contents vi

1 Introduction **1**
1.1 Kleene Algebra 1
1.2 Dynamic Logic 2
1.3 Program Schematology 3
1.4 KA and KAT 5
1.5 Thesis Overview 7

2 Kleene Algebra With Extra Equations **10**
2.1 Introduction 10
2.2 Preliminaries and a Negative Result 14
2.3 String Rewriting Systems 18
2.4 Completeness: Monoid Equations 28
2.5 Completeness: Partial Monoid Equations 33
2.6 Completeness: Typed Monoid Equations 39
2.7 Applications 46
2.7.1 Kleene algebra with tests 46
2.7.2 KAT with simple Hoare hypotheses 48
2.7.3 Redundant actions 49
2.7.4 Mutable tests 50
2.7.5 NetKAT 54
2.7.6 Test commutes with action 55
2.8 Conclusion 55

3 KAT With Extra Mutable Tests **57**
3.1 Introduction 57
3.2 Mutable Tests 60
3.2.1 Free KAT Generated by Mutable Tests 64
3.3 The Commutative Coproduct of KATs 69
3.4 KAT and Extra Mutable Tests 77
3.4.1 A Folk Theorem of Program Schematology 79
3.5 Conclusion 89

4 Kleene Algebra and Theories of Fixpoints **91**
4.1 Introduction 91
4.2 Nondeterministic Structure from Closure Operators 95
4.3 Continuity, Least Fixpoints and Nondeterminism 109
4.4 Typed Kleene Algebra With Products 120

4.5 The Theory of Parametric Fixpoints 125
4.6 Related Work 128
4.7 Conclusion 131

5 Conclusion **133**

Bibliography **134**

CHAPTER 1

INTRODUCTION

1.1 Kleene Algebra

Kleene algebra (KA) is an algebraic system that captures axiomatically the properties of a wide range of structures that arise in computer science and logic. It is named after Stephen Cole Kleene, who invented regular expressions and proved their equivalence to finite automata in his seminal work from the 1950's [58, 59]. Kleene algebra is the algebraic theory of these fundamental structures, but (as we will discuss later) it is not restricted only to these interpretations. The name *Kleene algebra* was coined by John H. Conway, who developed significantly in his monograph [31] the algebraic theory.

Already in the aforementioned work of Kleene [58, 59] the issue of axiomatization is touched upon. More specifically, Kleene identifies in his paper several valid equivalences that can be used for "algebraic transformations of regular expressions". The notion of equivalence he refers to is that of language equivalence, that is, two expressions are considered to be equivalent if they denote the same regular set of words.

Kleene algebra has many more natural interpretations besides the standard language-theoretic one. Kleene algebras arise in the context of relational algebra [97, 124], as well as in the context of the standard relation-theoretic semantics of imperative computation (based on binary relations on a state space) and the respective logics of programs [106, 107, 108, 65, 109]. Many more models can be found in the areas of automata and formal language theory (see, for example,

[89] for a generalized treatment using semirings and formal power series).

Interestingly, Kleene algebras also arise in the design and analysis of algorithms. They offer a convenient level of abstraction at which one can describe generic algorithms for computing reachability (i.e., reflexive transitive closure) and shortest paths in directed graphs [86, 5, 95, 37]. For the case of reachability, the algebra of $n \times n$ Boolean matrices is considered. For computing all-pairs shortest paths, the so called $(\min, +)$ algebra is used. A Kleene algebra of convex polygons, with an operation for the vector sum of two polygons and an operation for the convex hull of the union of two polygons, has been used to solve a cycle problem in directed graphs [56].

1.2 Dynamic Logic

Kleene algebra forms an essential component of Propositional Dynamic Logic (PDL) [35, 36], which is a propositionally abstracted logical system that can describe partial correctness, equivalence, and termination. PDL is the propositional counterpart of Pratt's Dynamic Logic (DL) [106, 50, 51] (see also [47, 83, 48, 49] for general references). The main idea is to integrate programs in an assertion language that combines Boolean logic and modal operators [15]. If f is a program and p is an assertion, then the expression $[f]p$ is a new assertion whose meaning is that "whenever f is executed, the assertion p holds upon termination." So, for every program f we have a "box" modal operator $[f]$. A dual "diamond" modal operator $\langle f \rangle$ is given by $\langle f \rangle p = \neg[f]\neg p$. The meaning of $\langle f \rangle p$ is that "the program f can terminate in a state satisfying p."

Kleene algebra can be regarded as an equational subsystem of PDL. For practical applications in program verification, many simple program transformations do not require the full power of PDL, but can be modeled in a purely equational system using the axioms of KA. We note that the Boolean algebra component of PDL is essential for programs, because it is necessary for modeling the guards of conditionals and while loops. A combination of Kleene algebra and Boolean algebra was proposed in [71, 72], which is particularly convenient for reasoning about conventional imperative programming constructs such as conditionals and while loops. This variant of KA, which we will define later, is called *Kleene algebra with tests* (KAT) and it subsumes traditional approaches to program verification such as Floyd-Hoare logic [38, 52, 53, 32, 8, 9, 75].

1.3 Program Schematology

A *program schema* is an abstract algorithm or program, where the meaning of the primitive operations is left unspecified. A program schema is also called an *uninterpreted* program, because the base functions lack a fixed interpretation. The theory of program schematology has a rich history that goes back to the early work of Ianov [55], which is also reported by Rutledge in [115]. One of the goals of this line of research is to study problems like program equivalence and partial correctness (which are undecidable for general programs) in an abstract setting. The hope is that these unsolvable problems might become solvable for such abstract models of computation.

The program schemes of Ianov represent only the sequential and control properties of iterative programs, and they disregard almost all information about the nature of the primitive operations. This massive abstraction sacrifices a large part of the essential structure of programs, and it leaves us with a model of computation that is little more than finite automata [111]. The benefit, however, is that a decision procedure for the equivalence of such schemes can be obtained.

An enormous amount of work has appeared on schematic models of computation that extend Ianov's model with more features, see for example [99, 100, 93, 33, 57, 90, 102, 101, 40, 41, 26, 11, 94, 110]. In Ianov's model the state space of a program is seen as one indivisible entity and the program actions are modeled as unary functions that act on the entire program space. Unfortunately, even the seemingly innocuous extension of the schematic language to allow several distinct program variables $x, y, z, \ldots$ that subdivide the program space and uninterpreted actions that can read from and assign to variables individually (e.g., an assignment $y \leftarrow f(x, z)$ that only changes the value of y) gives rise to undecidable reasoning problems [90, 110]. This devastating negative result meant that it was fundamentally impossible to realize the goal of obtaining decision procedures for reasoning about rich uninterpreted models. As a result, the area of program schematology focused largely on questions related to expressiveness and translatability between different models [102, 122, 123, 121, 30, 26], thus illuminating the relative power of programming features. More recent papers that essentially study partial-correctness properties for propositional program schemes (with some extensions) are [84, 91, 92].

Program schemes are typically presented as directed graphs, where the vertices are labeled with atomic actions (e.g., variable assignments) or tests, and the edges represent the flow of control. Unfortunately, this formalism is not compositional, and typical arguments for establishing scheme equivalence involve complicated "surgery" on graphs [94]. Kleene algebra, which replaces combinatorial reasoning on graphs by algebraic manipulation [7, 44], can offer a much more satisfying approach to the problem of scheme equivalence.

1.4 KA and KAT

A Kleene algebra is an algebraic structure $(K, +, \cdot, ^*, 0, 1)$ that consists of a nonempty set K, together with distinguished binary operations $+$ (sum) and $\cdot$ (multiplication), a unary operation *, and constants 0 and 1 that satisfy certain properties. An important example of a Kleene algebra is the family $\mathsf{Reg}(\Sigma)$ of regular sets over a finite alphabet Σ with the operations $\cup$, $\cdot$, *, $\emptyset$ and $\{\varepsilon\}$, where

$$A \cdot B = \{xy \mid x \in A \text{ and } y \in B\} \qquad A^0 = \{\varepsilon\}$$

$$A^* = \textstyle\bigcup_{n \geq 0} A^n \qquad A^{n+1} = A^n \cdot A$$

We write ε above to denote the empty string. The equational theory of $\mathsf{Reg}(\Sigma)$, which was first studied by Kleene [58, 59], is called the *algebra of regular events*.

The problem of finding a "good" axiomatization for the algebra of regular events has a long and rich history. The finitary axiomatic systems of Salomaa [118] are sound and complete for the equational theory of $\mathsf{Reg}(\Sigma)$, but they contain special rules of inference that are unsound for other natural interpretations. It was shown by Redko that the algebra of regular events can have no finite equational axiomatization [113]. A refinement of this negative result to the case of a unary alphabet was presented in [2]. Conway contributed significantly to the development of the algebraic theory in his monograph [31], but his treatment is mostly infinitary. A finitary complete axiomatization involving only equations and equational implications was presented in [69, 70] by Kozen. This axiomatization has the advantage over Salomaa's that it is sound for several important interpretations in addition to the standard one. The axiomatizations by Krob [87, 88] and Bloom-Ésik [16] involve infinitely many equations given schematically.

Throughout this thesis, the notion of Kleene algebra that we use will be that of Kozen [69, 70]. So, a *Kleene algebra* is an algebraic structure $(K, +, \cdot, ^*, 0, 1)$ that is a model of the equations and equational implications of Figure 1.1, where we write $x \leq y$ to abbreviate $x + y = y$. All axioms are implicitly universally quantified. We often elide the operation $\cdot$ and write xy instead of $x \cdot y$. The axioms of Figure 1.1 that do not involve * are those of *idempotent semirings*. The rest of the axioms say, informally, that * behaves like the Kleene star operator of language theory or the reflexive transitive operator on relations. Three useful properties that hold in all Kleene algebras are:

$$\textit{sliding rule}: \quad x(yx)^* = (xy)^*x \tag{1.1}$$

$$\textit{denesting rule}: \quad (x+y)^* = x^*(yx^*)^* \tag{1.2}$$

$$\textit{bisimulation rule}: \quad xy = yz \Rightarrow x^*y = yz^* \tag{1.3}$$

The main result of [69, 70] is that the axiomatization of Figure 1.1 is sound and complete for the algebra of regular events. That is, two regular expressions e and f over Σ denote the same regular set in $\mathsf{Reg}(\Sigma)$ iff the equation $e = f$ is a logical consequence of the axioms. An equivalent way to state this result is that $\mathsf{Reg}(\Sigma)$ is the free Kleene algebra on generators Σ. This major completeness result has recently been formalized in the Coq proof assistant [24, 25].

Motivated from applications to program verification, a combination of Kleene algebra with Boolean algebra was presented in [71, 72]. A *Kleene algebra with tests* (or KAT) is a two-sorted algebra $(K, B, +, \cdot, ^*, 0, 1, \neg)$ with carriers

$$\begin{aligned}(x+y)+z &= x+(y+z) & (xy)z &= x(yz)\\ x+0 &= x & 1x &= x\\ x+y &= y+x & x1 &= x\\ x+x &= x & &\end{aligned}$$

$$\begin{aligned}0x &= 0 & 1+xx^* &\le x^*\\ x0 &= 0 & 1+x^*x &\le x^*\\ x(y+z) &= xy+xz & xy \le y &\Rightarrow x^*y \le y\\ (x+y)z &= xz+yz & yx \le y &\Rightarrow yx^* \le y\end{aligned}$$

Figure 1.1: Axiomatization of Kleene algebras.

$B \subseteq K$ and $\neg : B \to B$ such that the reduct $(K, +, \cdot, ^*, 0, 1)$ is a Kleene algebra and $(B, +, \cdot, 0, 1, \neg)$ is a Boolean algebra. The elements of B are called *tests*. The operation $\neg$ is called *negation* and we also write $\bar{p}$ to mean $\neg p$. The familiar programming constructs of *sequential composition*, *conditionals*, and *while loops* can be modeled in KAT as follows:

$$f;g = fg \qquad \textsf{if}\ p\ \textsf{then}\ f\ \textsf{else}\ g = pf + \bar{p}g \qquad \textsf{while}\ p\ \textsf{do}\ f = (pf)^*\bar{p}$$

There is a semantic justification of the above encodings using the standard relation-theoretic semantics of imperative programs [83].

1.5 Thesis Overview

The work presented in this thesis builds upon previous completeness results on Kleene algebra [69, 70] and Kleene algebra with tests [71, 72, 82]. We study variations and extensions of Kleene algebra that are useful for reasoning about computer programs and other structures of computational interest. The results that we present attest to the versatility of KA and KAT, and confirm that classical equational reasoning is appropriate for a wide range of verification tasks. The technical material that follows in Chapters 2, 3 and 4 is largely based on the publications [80, 44, 79].

In Chapter 2 we investigate extensions of KA and KAT with extra equational assumptions. The purpose of such extensions in the context of program verification is to capture some properties of the domain of computation that are necessary for a given verification task. In more technical terms, we identify sufficient conditions for the construction of free language models for systems of Kleene algebra with additional equations. The construction applies to a broad class of extensions of KA and provides a uniform approach to deductive completeness. Our theorem is general enough to give as easy corollaries numerous known and new completeness results.

Chapter 3 explores extensions of KAT with extra structure that enables classical constructions from the field of program schematology. It is known that certain program transformations require a small amount of mutable state, a feature not explicitly provided by KAT. In this paper we show how to axiomatically extend KAT with this extra feature in the form of *mutable tests*. The extension is conservative and is formulated as a general commutative coproduct construction. We give several results on deductive completeness of the system, as well as a significant example illustrating its use.

In Chapter 4 we develop a typed equational system that subsumes both iteration theories (see the work of Bloom and Ésik [17]) and typed Kleene algebra [74] in a common framework. Our approach is based on categories with binary products endowed with extra structure to handle nondeterminism. We show that our typed variant of Kleene algebra extends conservatively the general "(in)equational theory of parametric fixpoints," which is characterized by the standard CPO model.

Each of the Chapters 2, 3 and 4 is self-contained and can be read independently of the rest.

CHAPTER 2

KLEENE ALGEBRA WITH EXTRA EQUATIONS

2.1 Introduction

Kleene algebra (KA) is the algebra of regular expressions. Introduced by Stephen Cole Kleene in 1956, it is fundamental and ubiquitous in computer science. It has proven useful in countless applications, from program specification and verification to the design and analysis of algorithms [7, 14, 27, 28, 29, 72, 81, 6].

One can augment KA with Booleans in a seamless way to obtain Kleene algebra with tests (KAT). Unlike many other related logics for program verification, KAT is classically based, requiring no specialized syntax or deductive apparatus other than classical equational logic. In practice, statements in the logic are typically universal Horn formulas

$$s_1 = t_1 \to s_2 = t_2 \to \cdots \to s_n = t_n \to s = t,$$

where the conclusion $s = t$ is the main target task and the premises $s_i = t_i$ are the verification conditions needed to prove it. The conclusion $s = t$ may encode a partial correctness assertion, an equivalence between an optimized and an unoptimized version of a program, or an equivalence between a program annotated with static analysis information and the unannotated program. The verification conditions $s_i = t_i$ are typically simple properties of the underlying domain of computation that describe how atomic actions interact with atomic assertions. They may require first-order interpreted reasoning, but are proven once and for all, then abstracted to propositional form. The proof of the conclusion $s = t$ from the premises takes place at the propositional level in KAT. This methodology affords a clean separation of the theory of the domain of computation from the program restructuring operations. It is advantageous to separate the two levels of reasoning, because the full first-order theory of the domain of computation may be highly undecidable, even though we may only need small parts of it. By isolating those parts, we can often maintain decidability and deductive completeness.

A typical form of premise that arises frequently in practice is a *commutativity condition* $pb = bp$ for an action p and a test b. This captures the idea that the action p does not affect the truth of b. For example, the action p might be an assignment $x := 3$ and b might be a test $y = 4$, where x and y are distinct variables. It is clear that the truth value of b is not affected by the action p, so it would be the same before as after. But once this is established, we no longer need to know what p and b are, but only that $pb = bp$. It follows by purely equational reasoning in KAT that $p_1 b = bp_1 \to \cdots \to p_n b = bp_n \to qb = bq$, where q is any program built from atomic actions $p_1, \ldots, p_n$.

In some instances, Horn formulas with premises of a certain form can be reduced to the equational theory without loss of deductive completeness or decision efficiency using a technique known as *elimination of hypotheses* [27, 82, 45]. One important class of premises for which this is possible are those of the form $s = 0$. The universal Horn theory restricted to premises of this form is called the *Hoare theory*, because it subsumes Hoare logic: the partial correctness assertion $\{b\}p\{c\}$ can be encoded as the equation $bp\bar{c} = 0$. Other forms that arise frequently in practice are $bp = b$, which says that the action p is not necessary if b is true, useful in optimizations to eliminate redundant actions; and $pq = qp$, which says that the atomic actions p and q can occur in either order with the same effect, useful in reasoning about concurrency. Unfortunately, KAT with general commutativity assumptions $pq = qp$ is undecidable [77].

As a case in point, the NetKAT system [6] incorporates a number of such equational premises as part of the theory, which are taken as additional axioms besides those of KAT. Proofs of deductive completeness and complexity as given in [6] required extensive adaptation of the analogous proofs for KA and KAT. Indeed, this was already the case with KAT, which was an adaptation of KA to incorporate an embedded Boolean algebra.

Although each of these instances was studied separately, there are some striking similarities. It turns out that the key to progress in all of them is the identification of a suitable class of *language models* that characterize the equational theory of the system. A language model is a structure in which expressions are interpreted as sets of elements of some monoid. The language models should form the free models for the system at hand. For KA, a language model is the regular sets of strings over a finite alphabet, elements of a free monoid; for KAT, the regular sets of guarded strings; for NetKAT, the regular sets of strings of a certain reduced form. Once a suitable class of language models can be determined, this opens the door to a systematic treatment of deductive completeness. It is also clear from previous work [6, 39, 44, 114, 22] that the existence of coalgebraic decision algorithms also depends strongly on the existence of language models (although we do not develop this connection here). The question thus presents itself: Is there a general set of criteria that admit a uniform construction of language models and that would apply in a broad range of situations and subsume previous ad hoc constructions? That is the subject of this chapter.

Alas, such a grand unifying framework is unlikely, given the negative results of [77] and of §2.2. However, we have identified a framework that goes quite far in this direction. It applies in the case in which the additional equational axioms are monoid equations or partial monoid equations (as is the case in all the examples mentioned above) and is based on a well-studied class of rewrite systems called *inverse context-free systems* [23]. We give criteria in terms of these rewrite systems that imply the existence of free language models in a wide range of previously studied instances, as well as some new ones.

This chapter is organized as follows. In §2.2 we present preliminary definitions and our negative result limiting the applicability of the method. In §2.3 we establish a connection between the classical theory of string rewriting and Kleene algebra. We recall from [23] the definition of *inverse context-free rewrite systems* and the key result that they preserve regularity. The original proof of [23] involved an automata-theoretic construction, but we show that it can be carried out axiomatically in KA. In §2.4 and §2.5 we give examples of total and partial monoid equations and give a general construction that establishes completeness in those cases. These constructions are special cases of the more general results of §2.6, but we start with them as a conceptual first step to illustrate the ideas. However, we can already derive some interesting consequences in these special cases. In §2.6, we establish completeness for typed monoid equations. This is the most general setting covered here. We give the completeness proof in §2.6, along with several applications in §2.7. In §2.8 we present conclusions, future work, and open problems.

2.2 Preliminaries and a Negative Result

A *monoid* is an algebraic structure $(M, \cdot, 1)$, where the *multiplication* operation $\cdot$ is associative and 1 is a left and right *unit* for multiplication. That is, a monoid M satisfies

$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \qquad\qquad 1 \cdot x = x \qquad\qquad x \cdot 1 = x$$

for all elements x, y, z in M. For a subset $S \subseteq M$, define $\langle S \rangle$ to be the smallest subset of M that contains S and 1 and is closed under multiplication. We say that S is a *generator* of M (or that S *generates* M) if $M = \langle S \rangle$. We say that M is *finitely generated* if it has a finite generator. For a set Σ of symbols, we write Σ^* for the set of words (strings) over Σ. The monoid $(\Sigma^*, \cdot, \varepsilon)$ consists of the carrier Σ^*, together with the operation $\cdot$ of string concatenation and the empty string ε as unit. We say that $(\Sigma^*, \cdot, \varepsilon)$ is the *free monoid* generated by Σ. A more general method of defining a monoid is by a *presentation*. For a presentation we specify a set Σ of *generators* and a binary relation $R \subseteq \Sigma^* \times \Sigma^*$ on strings. We also write a pair (u, u') in R as an equation $u \equiv u'$. Define $\equiv_R$ to be the smallest congruence (equivalence relation and congruence with respect to concatenation) of Σ^* that contains R. The congruence class of a string u is

$$[u]_R = \{v \in \Sigma^* \mid u \equiv_R v\}.$$

Now, define

$$M = \langle \Sigma \mid R \rangle = \Sigma^*/R$$

to be the monoid whose carrier is the set $\{[u]_R \mid u \in \Sigma^*\}$ of $\equiv_R$-congruence classes. Multiplication is given by $[u]_R \cdot [v]_R \mapsto [uv]_R$, and the unit is $[\varepsilon]_R$. We say that $\langle \Sigma \mid R \rangle$ is the (presented) monoid with generators Σ and equations R. If Σ and R are finite, we say that they constitute a finite presentation, and that $\langle \Sigma \mid R \rangle$ is a finitely presented monoid.

Assumption 1 (Finite Alphabet)**.** For the rest of the chapter, even when it is not explicitly noted, we will be assuming implicitly that an alphabet Σ of letters is finite.

Example 2 (Commuting Letters)**.** Let $\Sigma = \{a, b\}$ be a finite alphabet with letters a and b. We consider only the commutativity equation $ab \equiv ba$. The finitely presented monoid

$$M = \langle a, b \mid ab \equiv ba \rangle$$

is isomorphic to the monoid $\mathbf{N} \times \mathbf{N}$, where $\mathbf{N}$ is the set of natural numbers, with multiplication $(m_a, m_b), (n_a, n_b) \mapsto (m_a + n_a, m_b + n_b)$ and unit $(0, 0)$. The mapping $h : M \to \mathbf{N} \times \mathbf{N}$ is the unique monoid homomorphism given by $h(a) = (1, 0)$ and $h(b) = (0, 1)$. It is also surjective and injective, thus witnessing the claimed isomorphism.

Definition 3 (Regular Expressions and Language Interpretation)**.** We define *regular expressions* over the finite alphabet Σ to be the terms given by the grammar

$$e ::= a \in \Sigma \mid 1 \mid 0 \mid e + e \mid e; e \mid e^*.$$

We can interpret a regular expression as a subset of a monoid $M = \langle \Sigma \mid R \rangle$ with multiplication $\cdot$ and identity $1_M = [\varepsilon]_R$. The function $\mathcal{R}_M$, called the *language interpretation* in M, sends a regular expression to a set of elements of M:

$$\begin{aligned} \mathcal{R}_M(a) &= \{[a]_R\} & \mathcal{R}_M(e_1 + e_2) &= \mathcal{R}_M(e_1) \cup \mathcal{R}_M(e_2) \\ \mathcal{R}_M(1) &= \{1_M\} & \mathcal{R}_M(e_1; e_2) &= \mathcal{R}_M(e_1) \cdot \mathcal{R}_M(e_2) \\ \mathcal{R}_M(0) &= \emptyset & \mathcal{R}_M(e^*) &= \textstyle\bigcup_{n \geq 0} \mathcal{R}_M(e)^n \end{aligned}$$

where $\cdot$ is lifted to subsets of M as $A \cdot B = \{u \cdot v \mid u \in A, v \in B\}$, and the n-fold product A^n is defined inductively as $A^0 = \mathcal{R}_M(1)$ and $A^{n+1} = A^n \cdot A$.

The image of the interpretation $\mathcal{R}_M$ together with the operations $\cup$, $\cdot$, *, $\emptyset$, $\{1_M\}$ is the *algebra of regular sets* over M, denoted by $\mathsf{Reg}(M)$. If M is the free monoid Σ^*, then $\mathcal{R}_M$ is the standard language interpretation $\mathcal{R}$ of regular expressions.

Example 4. As in Example 2, consider the two-element alphabet $\Sigma = \{a, b\}$ and the monoid $M = \langle a, b \mid ab \equiv ba \rangle$. The language (in M) denoted by the expression $a; (b; a)^*$ is

$$\mathcal{R}_M(a; (b; a)^*) = \{[a(ba)^n] \mid n \geq 0\} = \{[a^{n+1}b^n] \mid n \geq 0\}.$$

We have made use of the fact that $a(ba)^n \equiv a^{n+1}b^n$, which can be shown using $ab \equiv ba$.

It is known that the algebra of regular sets $\mathsf{Reg}(\Sigma^*)$ is the free Kleene algebra generated by Σ [70]. This is equivalent to the completeness of the axioms of KA for the standard language interpretation $\mathcal{R}$ of regular expressions. That is, for any two regular expressions e_1, e_2 over Σ, if $\mathcal{R}(e_1) = \mathcal{R}(e_2)$ then $\mathsf{KA} \vdash e_1 \equiv e_2$. The question then arises if this result extends to the general case of $\mathsf{Reg}(M)$ for a (finitely) presented monoid $M = \langle \Sigma \mid R \rangle$. We ask the question of whether $\mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)$ implies provability of $e_1 \equiv e_2$ in a system of KA augmented with (at least) the equations corresponding to R.

In general, the answer to the question posed in the previous paragraph is negative. That is, there exists a finitely presented monoid $M = \langle \Sigma \mid R \rangle$ such that the equational theory of $\mathsf{Reg}(M)$ is not recursively enumerable, and therefore not recursively axiomatizable. The *equational theory* of the Kleene algebra $\mathsf{Reg}(M)$ is the set of equations between regular expressions that are true in $\mathsf{Reg}(M)$ under the interpretation $\mathcal{R}_M$, i.e., the set

$$\{e_1 \equiv e_2 \mid \mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)\}.$$

We show this negative result using the ideas developed in [73, 77]. The proof specifies a way to construct effectively the monoid whose existence we claim.

Theorem 5. There exists a finitely presented monoid M such that the equational theory of $\mathsf{Reg}(M)$ is not recursively enumerable.

Proof. We define the Turing machine $\mathcal{M}_T$, which takes as input a pair of natural numbers (n, u). The number n is interpreted as the index of a Turing machine $\mathcal{M}_n$, and the number u is meant to be given as input to the machine $\mathcal{M}_n$. In order to encode (n, u) as a string, we take as input alphabet the set $\Sigma = \{a, \#\}$. The pair (n, u) is encoded as the string $a^n\#a^u$. We describe now the algorithm that $\mathcal{M}_T$ implements. Let x be the input string.

1. If the input string x is not of the appropriate form $a^n\#a^u$ then halt.
2. From the index n compute the description of the Turing machine $\mathcal{M}_n$.
3. Simulate the execution $\mathcal{M}_n(u)$ of the machine $\mathcal{M}_n$ on input u. If the computation $\mathcal{M}_n(u)$ halts, then erase the tape (by filling it with blank symbols) and halt at a special halting state.

Recall now the totality problem $\text{TOTAL} = \{n \mid \mathcal{M}_n \text{ halts on every input}\}$, which is known to be Π^0_2-complete, and observe the equivalence:

$$n \in \text{TOTAL} \iff \mathcal{M}_T(n, u) \text{ halts for all } u \geq 0.$$

It is shown in [73, 77] that for every Turing machine $\mathcal{M}$, there exists a finitely presented monoid $M = \Delta^*/E$, which intuitively encodes the computations of the machine. For every input string x there exists an effectively computable equation $e_1; x; e_2 \equiv e$ such that $\mathcal{M}$ halts on input x iff $\Delta^*/E \models e_1; x; e_2 \equiv e$. All expressions e_1, e_2, e are strings.

Suppose now that the monoid $M = \Delta^*/E$ is the one corresponding to the machine $\mathcal{M}_T$ described in the previous paragraph. The index n belongs to TO-

TAL iff

$$\mathsf{Reg}(M), \mathcal{R}_M \models e_1; a^n\#a^u; e_2 \equiv e, \text{ for all } u \geq 0 \iff$$

$$\mathsf{Reg}(M), \mathcal{R}_M \models e_1; a^n\#a^u; e_2 \leq e, \text{ for all } u \geq 0 \iff$$

$$\mathsf{Reg}(M), \mathcal{R}_M \models e_1; a^n\#a^*; e_2 \leq e.$$

The last statement says that the equation $e_1; a^n\#a^*; e_2 \leq e$ belongs to the equational theory of $\mathsf{Reg}(\Delta^*/E)$. It follows that this equational theory is Π^0_2-hard and therefore not recursive enumerable. $\square$

This negative result says that we can only hope to identify subclasses of monoid presentations $M = \langle \Sigma \mid R \rangle$ such that the algebra $\mathsf{Reg}(M)$ of regular sets over M is axiomatizable. The idea is to first restrict attention to those monoid presentations for which the equations can be oriented to give a confluent and terminating rewrite system. This allows one to consider as canonical representatives the irreducible strings of the congruence classes. Then, we focus on a subclass that allows two crucial algebraic constructions: a "descendants" automata-theoretic construction, and an "ancestors" construction.

Note that Theorem 5 is a strengthening of [77, Theorem 4.1(ii)]. The theorem of [77] gives a uniform Π^0_2-lower bound when the monoid is considered part of the input, whereas Theorem 5 gives a Π^0_2-lower bound for the theory of a fixed monoid.

2.3 String Rewriting Systems

In this section we establish a connection between the classical theory of string rewriting systems and Kleene algebra. More specifically, we recall a result regarding the preservation of regularity: for every *inverse context-free* system R and a regular set L, the set of the R-descendants of L is also regular [23]. This result involves an automata-theoretic construction, which can be modeled in KA, because an automaton can be represented as an appropriate KA term [31, 70]. We show here that the combinatorial arguments of the construction can then be replaced by equational reasoning in KA. As it turns out, this connection will allow us to obtain powerful completeness metatheorems in later sections.

A *string rewriting system* R over a finite alphabet Σ consists of rewrite rules $\ell \to r$, where ℓ and r are finite strings over Σ. This extends to the *one-step rewrite relation* $\to_R$, given by $x\ell y \to_R xry$, for strings x, y and rule $\ell \to r$ of R. If $x \to_R y$ then we say that y is an *R-successor* of x, and x is an *R-predecessor* of y. We write $\to_R^*$ for the reflexive-transitive closure of $\to_R$, which is called the *rewrite relation* for R. If u, v are strings for which $u \to_R^* v$ we say that v is an *R-descendant* of u, and that u is an *R-ancestor* of v. For a set of strings L we put:

$$\mathsf{Desc}_R(L) = \{v \mid \exists u \in L.\ u \to_R^* v\}$$

$$\mathsf{Ance}_R(L) = \{u \mid \exists v \in L.\ u \to_R^* v\}$$

So, $\mathsf{Desc}_R(L)$ is the set of all the R-descendants of the strings in L, and similarly $\mathsf{Ance}_R(L)$ is the set of all R-ancestors of the strings in L. The *inverse system* R^{-1} of R is the system that results by taking a rule $r \to \ell$ for every rule $\ell \to r$ of R. If u is an R-ancestor of a string v, then u is an R^{-1}-descendant of v. We define the symmetric relation

$$\leftrightarrow_R = \{(xuy, xvy) \mid u \to v \text{ or } v \to u \text{ is an } R\text{-rule}\}.$$

We write $\leftrightarrow_R^*$ for the reflexive transitive closure of $\leftrightarrow_R$. The relation $\leftrightarrow_R^*$ is an equivalence relation on Σ^*. In fact, it is a left and right congruence, because it satisfies additionally for all strings u, v, x, y: $u \leftrightarrow_R^* v$ implies that $xuy \leftrightarrow_R^* xvy$.

Equivalently, we can define $\leftrightarrow_R^*$ to be the smallest congruence on Σ^* that contains $\{(u, v) \mid u \to v \text{ is an } R\text{-rule}\}$. The congruence class of a string u is denoted by $[u]$.

Remark 6. Let R be a string rewrite system over Σ that has rules of the form $a \to r$, where $a \in \Sigma$ is a single letter, as well as rules of the form $\varepsilon \to r$. Let xy be a string. We claim that every R-descendant of xy is of the form uv, where u (resp. v) is an R-descendant of x (resp. y). This claim can be expressed equivalently with the equation

$$\mathsf{Desc}_R(xy) = \mathsf{Desc}_R(x) \cdot \mathsf{Desc}_R(y).$$

We can then prove its generalization $\mathsf{Desc}_R(L \cdot L') = \mathsf{Desc}_R(L) \cdot \mathsf{Desc}_R(L')$ to sets of strings.

Lemma 7. Let R be a rewrite system consisting of rules of the form $\varepsilon \to r$ and $a \to r$, where a is a letter. Assume further that all sets $\mathsf{Desc}_R(\varepsilon)$ and $\mathsf{Desc}_R(a)$ are regular with

$$\mathcal{R}(e_\varepsilon) = \mathsf{Desc}_R(\varepsilon) \qquad\qquad \mathcal{R}(e_a) = \mathsf{Desc}_R(a)$$

for some regular expressions e_ε and e_a. Consider the substitution θ, defined inductively by

$$\begin{aligned} \theta(a) &= e_a & \theta(e_1 + e_2) &= \theta(e_1) + \theta(e_2) \\ \theta(1) &= e_\varepsilon & \theta(e_1; e_2) &= \theta(e_1); \theta(e_2) \\ \theta(0) &= 0 & \theta(e^*) &= e_\varepsilon + \theta(e)^* \end{aligned}$$

Then, $\mathsf{Desc}_R(\mathcal{R}(e)) = \mathcal{R}(\theta(e))$ for every regular expression e. For the particular case where $\mathsf{Desc}_R(\varepsilon) = \{\varepsilon\}$, we can simplify the substitution by putting $\theta(e^*) = \theta(e)^*$.

Proof. The proof is by induction on the structure of e. We only give the argument for the case of e^*, because it is the most interesting one. First, we claim that

$$\mathsf{Desc}_R(\mathcal{R}(e)^n) = \mathcal{R}(\theta(e))^n \text{ for every } n > 0.$$

This is shown by induction on n, using the property of Remark 6. Now, we have for e^*:

$$\begin{aligned}
\mathsf{Desc}_R(\mathcal{R}(e^*)) &= \mathsf{Desc}_R(\{\varepsilon\} \cup \textstyle\bigcup_{n>0}\mathcal{R}(e)^n) && [\text{def. of } \mathcal{R}] \\
&= \mathsf{Desc}_R(\varepsilon) \cup \textstyle\bigcup_{n>0}\mathsf{Desc}_R(\mathcal{R}(e)^n) && [\text{def. of } \mathsf{Desc}_R] \\
&= \mathcal{R}(e_\varepsilon) \cup \textstyle\bigcup_{n>0}\mathcal{R}(\theta(e))^n && [\text{hyp. \& claim}] \\
&= \mathcal{R}(e_\varepsilon) \cup \{\varepsilon\} \cup \textstyle\bigcup_{n>0}\mathcal{R}(\theta(e))^n && [\varepsilon \in \mathcal{R}(e_\varepsilon)] \\
&= \mathcal{R}(e_\varepsilon) \cup \mathcal{R}(\theta(e)^*), && [\text{def. of } \mathcal{R}]
\end{aligned}$$

which is equal to $\mathcal{R}(e_\varepsilon + \theta(e)^*) = \mathcal{R}(\theta(e^*))$. In the fourth equality above we have made use of the fact that $\varepsilon \rightarrow_R^* \varepsilon$, which implies that $\varepsilon \in \mathsf{Desc}_R(\varepsilon) = \mathcal{R}(e_\varepsilon)$. □

Example 8. Let R be the rewrite system over $\Sigma = \{a, b\}$ that consists of the single rewrite rule $a \to aa$. The set $\mathsf{Desc}_R(a) = \{a^n \mid n > 0\}$ is regular and equal to $\mathcal{R}(e_a)$, where $e_a = a; a^*$. Clearly, $\mathsf{Desc}_R(b) = \{b\}$ is also regular and we put $e_b = b$. We consider the (simplified) substitution θ of Lemma 7, which gives us

$$\theta(a; (b; a)^*) = a; a^*; (b; a; a^*)^*.$$

Lemma 7 now says that $\mathsf{Desc}_R(\mathcal{R}(a; (b; a)^*)) = \mathcal{R}(a; a^*; (b; a; a^*)^*)$.

Example 9. Suppose that the rewrite system R over $\Sigma = \{a, b\}$ consists of the rewrite rule $\varepsilon \to aa$. We observe that the set $\mathsf{Desc}_R(\varepsilon) = \{(aa)^n \mid n \geq 0\}$ is regular and equal to $\mathcal{R}(e_\varepsilon)$, where $e_\varepsilon = (a; a)^*$. Moreover, we see that

$$\mathsf{Desc}_R(a) = \{a(aa)^n \mid n \geq 0\} \qquad \mathsf{Desc}_R(b) = \{(aa)^n b (aa)^n \mid n \geq 0\}$$

are regular. We put $e_a = a; e_\varepsilon$ and $e_b = e_\varepsilon; b; e_\varepsilon$. The hypotheses of Lemma 7 are satisfied.

Let R be a rewrite system. We say that R is *terminating* if there is no infinite rewrite chain $x_0 \to_R x_1 \to_R x_2 \to_R \cdots$. If R has rules of the form $\ell \to r$ with $|r| < |\ell|$ then it is terminating, because every rule application strictly reduces the length of the string. A string x is called *R-irreducible* if no rule of R applies to it, that is, there is no y with $x \to_R y$. We say that R is *confluent* if $u \to_R^* x$ and $u \to_R^* y$ imply that there exists z with $x \to_R^* z$ and $y \to_R^* z$. It is said that R has the *Church-Rosser property* (we also say that "R is Church-Rosser") if for all strings x, y with $x \leftrightarrow_R^* y$ there exists z such that $x \to_R^* z$ and $y \to_R^* z$. It is a standard result that confluence and the Church-Rosser property are equivalent [23]. A system R is said to be *locally (or weakly) confluent* if for all strings u, x, y with $u \to_R x$ and $u \to_R y$, there exists a string z such that $x \to_R^* z$ and $y \to_R^* z$. If R is both locally confluent and terminating, then R is confluent [23, 12].

Suppose that R is confluent and terminating. We map each string u to the unique R-irreducible string $\mathsf{nf}_R(u)$ that results from rewriting u as much as possible. For strings u, v, it holds that $u \leftrightarrow_R^* v$ iff $\mathsf{nf}_R(u) = \mathsf{nf}_R(v)$. So, two strings are congruent iff they can be rewritten to the same R-irreducible. For every congruence class $[u]$ of $\leftrightarrow_R^*$, we choose as *canonical representative* (normal form) the R-irreducible string $\mathsf{nf}_R(u)$.

Note 10. If R is a terminating rewrite system, then it is easy to see that the empty string ε is irreducible. Assume for contradiction that R has a rule $\varepsilon \to_R x$. Then we can obtain the infinite rewrite chain $\varepsilon \to_R x \to_R xx \to_R xxx \to_R \cdots$ which contradicts termination.

Definition 11 (Total Coalesced Product)**.** Let R be a confluent and terminating rewrite system over Σ, and I_R be the set of R-irreducible strings. Define the operation $\diamond$ on I_R by

$$u \diamond v = \mathsf{nf}_R(uv).$$

We call this operation the *coalesced product* with respect to R. We also lift the operation to sets of R-irreducible strings as $A \diamond B = \{u \diamond v \mid u \in A,\ v \in B\}$.

The structure $(I_R, \diamond, \varepsilon)$, as defined above, is a monoid. In fact, it is isomorphic to the monoid $\langle \Sigma \mid R \rangle$, where R in the presentation are the equations $\ell \equiv r$ corresponding to the rules $\ell \to r$ of the system.

Example 12. Let R be the rewrite system over $\Sigma = \{a, b\}$ with the single rule $ba \to ab$. The R-irreducible strings are $I_R = \{a^m b^n \mid m, n \geq 0\}$. The total coalesced product is

$$a^k b^l \diamond a^m b^n = \mathsf{nf}_R(a^k b^l a^m b^n) = a^{k+m} b^{l+n}.$$

The monoid $(I_R, \diamond, \varepsilon)$ is isomorphic to $\langle \Sigma \mid ab \equiv ba \rangle$.

Definition 13 ($\mathscr{C}$ and $\mathscr{G}$)**.** Let R be an arbitrary string rewrite system over the alphabet Σ. For a language $L \subseteq \Sigma^*$, we define

$$\mathscr{C}_R(L) = \bigcup_{u \in L} [u] = \{v \mid \exists u \in L.\ v \leftrightarrow_R^* u\}.$$

Assume additionally that R is confluent and terminating, so that the function nf_R is well-defined. For $L \subseteq \Sigma^*$, we define $\mathscr{G}_R(L) = \{\mathsf{nf}_R(u) \mid u \in L\}$.

The intuition for the above definitions is that the map $\mathscr{C}_R$ closes a set of strings under the congruence relation $\leftrightarrow_R^*$, and $\mathscr{G}_R$ reduces every string to its normal form (R-irreducible). We note that $\mathscr{C}_R(L)$, which is a set of strings, is *not* equal to $\{[u] \mid u \in L\}$, which is a set of equivalence classes of strings. Equivalently, we can define $\mathscr{C}_R(L)$ to be the smallest set that contains L and is closed under $\leftrightarrow_R^*$.

Lemma 14 ($\mathscr{C}$ and $\mathscr{G}$)**.** For a confluent and terminating rewrite system R over Σ:

1. $\mathscr{C}_R(L) = \mathscr{C}_R(\mathscr{G}_R(L))$, for a language $L \subseteq \Sigma^*$.

2. $\mathscr{G}_R(L) = \mathscr{G}_R(\mathscr{C}_R(L))$, for a language $L \subseteq \Sigma^*$.
3. $\mathscr{G}_R(L) = \mathscr{G}_R(L')$ iff $\mathscr{C}_R(L) = \mathscr{C}_R(L')$, for languages $L, L' \subseteq \Sigma^*$.
4. $\mathscr{C}_R(L) = \mathsf{Ance}_R(\mathsf{Desc}_R(L))$, for a language $L \subseteq \Sigma^*$.

Proof. For Part (1), we use the fact that $u \leftrightarrow_R^* \mathsf{nf}_R(u)$, which implies $[u] = [\mathsf{nf}_R(u)]$. So,

$$\mathscr{C}_R(L) = \bigcup\{[u] \mid u \in L\} = \bigcup\{[\mathsf{nf}_R(u)] \mid u \in L\}$$
$$= \bigcup\{[v] \mid v \in \mathscr{G}_R(L)\} = \mathscr{C}_R(\mathscr{G}_R(L)).$$

For Part (2), we first notice that $L \subseteq \mathscr{C}_R(L)$ and hence

$$\mathscr{G}_R(L) \subseteq \mathscr{G}_R(\mathscr{C}_R(L)) = \{\mathsf{nf}_R(u) \mid u \in \mathscr{C}_R(L)\}.$$

For the reverse containment, consider an arbitrary $u \in \mathscr{C}_R(L)$. We have to show that $\mathsf{nf}_R(u)$ is in $\mathscr{G}_R(L)$. Since $u \in \mathscr{C}_R(L)$, there exists some $u' \in L$ with $u \leftrightarrow_R^* u'$. It follows that $\mathsf{nf}_R(u) = \mathsf{nf}_R(u') \in \mathscr{G}_R(L)$. Part (3) is an immediate consequence of (1) and (2). Finally, the idea for Part (4) is that by closing L under R-descendants we obtain the normal forms. So, if we also close under R-ancestors we get the congruence class of every element of L. $\square$

A rewrite system R is said to *preserve regularity* if for every regular language L, the R-descendants $\mathsf{Desc}_R(L)$ form a regular set. A system R is called *inverse context-free* if it only contains rules of the form $\ell \to r$, where $|r| \leq 1$. That is, every right-hand side of a rule is either a single letter or the empty string. A classical result of the theory of string rewriting is that inverse context-free systems preserve regularity (see Chapter 4 of [23] for a detailed proof). The proof of this fact uses a construction on finite automata, which we briefly present here. We will be referring to it as the ***descendants construction***. Suppose that L is a regular language, recognized by an automaton $\mathcal{A}$. The automaton is possibly nondeterministic and it may have ε-transitions. We will describe a sequence of transformations on $\mathcal{A}$. When the sequence reaches a fixpoint, we obtain an automaton (nondeterministic with ε-transitions) that recognizes $\mathsf{Desc}_R(L)$.

- Suppose that the system R has a rule $\ell \to a$, where a is a single letter, and $\ell = \ell_1\ell_2 \cdots \ell_m$ is a string of length m. We assume that there is an ℓ-path from the state q_0 to the state q_n of the automaton. That is, a sequence

  $$q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} q_2 \xrightarrow{x_3} \cdots \xrightarrow{x_{n-1}} q_{n-1} \xrightarrow{x_n} q_n,$$

  where each x_i is a letter or ε, $x_1 \cdot x_2 \cdot \ldots \cdot x_n = \ell$, and each $q_{i-1} \xrightarrow{x_i} q_i$ is a transition of the automaton. We add the transition $q_0 \xrightarrow{a} q_n$. The idea is that if the automaton accepts the string $x\ell y$, then it should also accept the R-descendant xay.
- Similarly, suppose that the system R has a rule $\ell \to \varepsilon$, where ε is the empty string, and that there is an ℓ-path from the state q_0 to the state q_n. Then, we add the ε-transition $q_0 \xrightarrow{\varepsilon} q_n$ to the transition table of the automaton.

This process is iterated until no new transitions are added. We know that a fixpoint is always reached because there are only finitely many transitions that a finite automaton can have. The resulting automaton accepts exactly the set of R-descendants $\mathsf{Desc}_R(L)$.

Theorem 15 (The Descendants Construction Algebraically). Let R be an inverse context-free rewrite system and e a regular expression whose interpretation is $L = \mathcal{R}(e)$. We can construct effectively a new regular expression $\hat{e}$ s.t.

$$\mathsf{KA}_R \vdash e \equiv \hat{e} \quad \text{and} \quad \mathcal{R}(\hat{e}) = \mathsf{Desc}_R(L).$$

KA_R is the system KA augmented with an equation $\ell \equiv r$ for every rewrite rule $\ell \to r$ of R.

Proof. In [31] it is shown how to perform the construction of Kleene's theorem, which builds a finite-state automaton from a given regular expression, in terms of matrices. The automaton is possibly nondeterministic and may have ε-transitions, so for the expression e there is a matrix form $u; M^*; v$ with $\mathsf{KA} \vdash e \equiv u; M^*; v$, where u is a $1 \times n$ matrix, M is a $n \times n$ matrix, and v is a $n \times 1$ matrix. The matrix M is of the form

$$M = M(\varepsilon) + \textstyle\sum_a a \cdot M(a),$$

where a ranges over the alphabet Σ and $a \cdot -$ denotes scalar multiplication. Each $n \times n$ matrix $M(a)$ encodes the transitions of the automaton on input symbol a. The entries of $M(a)$ are either 0 or 1, hence the entries of $a \cdot M(a)$ are either 0 or a. Similarly, the entries of $M(\varepsilon)$ are either 0 or 1 and they give the ε-transitions of the automaton.

We will show in KA_R that for a transformation step from the automaton $u; M^*; v$ to the automaton $u; N^*; v$ we have that $\mathsf{KA}_R \vdash u; M^*; v \equiv u; N^*; v$. Suppose that $\ell \to a$ is a rule of R, $\ell = \ell_1 \ell_2 \cdots \ell_m$, and there is an ℓ-path from q_0 to q_n in the automaton:

$$q_0 \xrightarrow{x_1} q_1 \xrightarrow{x_2} \cdots \xrightarrow{x_{n-1}} q_{n-1} \xrightarrow{x_n} q_n,$$

with $x_1 \cdot x_2 \cdots x_{n-1} \cdot x_n = \ell$. Since each $q_{i-1} \xrightarrow{x_i} q_i$ is a transition of the automaton, we get

$$\mathsf{row}(q_{i-1}); M(x_i); \mathsf{col}(q_i) \equiv 1.$$

The above equation says that the (q_{i-1}, q_i)-indexed entry of $M(x_i)$ is equal to 1. We write $\mathsf{row}(q)$ for the row vector ($1 \times n$ matrix) that contains 1 at the q-indexed position and 0 in the rest of the positions. Similarly, $\mathsf{col}(q)$ is the column vector ($n \times 1$ matrix) with 1 at position q and 0 elsewhere. It is easy to see that $\mathsf{row}(q); \mathsf{col}(q) \equiv 1$, and $\mathsf{col}(q_i); \mathsf{row}(q_j)$ is equal to the $n \times n$ matrix with 1 at position

(q_i, q_j) and 0 elsewhere. So, the inequality

$$\mathsf{col}(q_{i-1}); \mathsf{row}(q_i) \leq M(x_i)$$

is another way of expressing the fact that $q_{i-1} \xrightarrow{x_i} q_i$ is a transition of the automaton. We define $N(a)$ so that $N(a) \equiv M(a) + \mathsf{col}(q_0); \mathsf{row}(q_n)$. This means that

$$N = M + a \cdot \mathsf{col}(q_0); \mathsf{row}(q_n).$$

Since $M \leq N$, it follows by monotonicity of * that $M^* \leq N^*$ and hence $u; M^*; v \leq u; N^*; v$.

Now, we have to show that $u; N^*; v \leq u; M^*; v$, which is implied by $N^* \leq M^*$. In order to make our exposition more understandable, we give the proof using a specific example. Suppose we have the rule $\ell \to a$, where $\ell = ab$, and the ℓ-path we consider is

$$q_0 \xrightarrow{a} q_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{b} q_3.$$

We add the transition $q_0 \xrightarrow{a} q_3$ to the automaton. So, $N(a) \equiv M(a) + \mathsf{col}(q_0); \mathsf{row}(q_3)$, and $N \equiv M + a \cdot \mathsf{col}(q_0); \mathsf{row}(q_3)$. Observe the provability of the following:

$$\begin{aligned} a \cdot \mathsf{col}(q_0); \mathsf{row}(q_3) &\equiv a \cdot \mathsf{col}(q_0); \mathsf{row}(q_1); \mathsf{col}(q_1); \mathsf{row}(q_2); \mathsf{col}(q_2); \mathsf{row}(q_3) \\ &\leq a \cdot M(x_1); M(x_2); M(x_3) \\ &\equiv a; b \cdot M(a); M(\varepsilon); M(b) \\ &\equiv aM(a); M(\varepsilon); bM(b) \\ &\leq M; M; M. \end{aligned}$$

In the second equation above we have used the axiom $ab \equiv a$, because the rule $ab \to a$ is in R. It follows that $N \leq M + M; M; M$, and therefore $N^* \leq (M + M; M; M)^* \leq M^*$.

If the original automaton form is $u; M_0^*; v$, the descendants construction gives us a finite sequence of matrix forms $u; M_0^*; v$, $u; M_1^*; v$, ..., $u; M_k^*; v$ with

$$\mathsf{KA}_R \vdash u; M_0^*; v \equiv u; M_1^*; v \equiv \cdots \equiv u; M_k^*; v.$$

No new transition can be added to the last automaton. So, the last automaton form of the sequence gives us all the descendants of $\mathcal{R}(e)$. That is,

$$\mathcal{R}(u; M_k^*; v) = \mathsf{Desc}_R(\mathcal{R}(u; M_0^*; v)) = \mathsf{Desc}_R(\mathcal{R}(e)),$$

because $\mathsf{KA} \vdash e \equiv u; M_0^*; v$. We put $\hat{e} = u; M_k^*; v$, and the proof is complete. $\square$

Theorem 15 says that the descendants construction, which is combinatorial, can be modeled algebraically in the system of KA with extra equations that correspond to the rules of the rewrite system. This is a central technical result that will be useful later for establishing our completeness theorems.

2.4 Completeness: Monoid Equations

In this section we present our first completeness metatheorem, from which we can prove the existence of free language models for systems of KA with extra monoid equations. Our metatheorem is not only a conceptual first step towards the more general partial monoid and typed monoid cases, which we investigate in §2.5 and §2.6 respectively, but it also allows us to obtain previously unknown completeness results. As a concrete novel application, think of the assignment statement $x := c$, where c is a constant. The action $x := c$ is idempotent, meaning that the effect of the program $x := c; x := c$ is the same as the effect of $x := c$. We express this fact with the monoid equation $aa \equiv a$, where a is a single-letter abstraction of the assignment. KA can be augmented with any number of such idempotence equations, and our metatheorem implies the existence of a free language model (see Example 21).

Definition 16 (Language Interpretations $\mathcal{G}$ and $\mathcal{C}$)**.** Let R be a confluent and terminating rewrite system. The corresponding coalesced product is $\diamond$. We define the function $\mathcal{G}_R$ that sends a regular expression to a set of R-irreducibles:

$$\begin{aligned} \mathcal{G}_R(a) &= \{\mathsf{nf}_R(a)\} & \mathcal{G}_R(e_1 + e_2) &= \mathcal{G}_R(e_1) \cup \mathcal{G}_R(e_2) \\ \mathcal{G}_R(0) &= \emptyset & \mathcal{G}_R(e_1; e_2) &= \mathcal{G}_R(e_1) \diamond \mathcal{G}_R(e_2) \\ \mathcal{G}_R(1) &= \{\varepsilon\} & \mathcal{G}_R(e^*) &= \textstyle\bigcup_{n \geq 0} \mathcal{G}_R(e)^{\langle n \rangle} \end{aligned}$$

where, for a set A of R-irreducibles, $A^{\langle n \rangle}$ is defined by $A^{\langle 0 \rangle} = \mathcal{G}_R(1)$ and $A^{\langle n+1 \rangle} = A^{\langle n \rangle} \diamond A$. We also define the interpretation $\mathcal{C}_R(e) = \mathscr{C}_R(\mathcal{R}(e)) = \bigcup_{u \in \mathcal{R}(e)} [u]$.

Remark 17. Let R be a confluent and terminating system over Σ, and $M = \langle \Sigma \mid R \rangle$ be the corresponding monoid. For a regular expression e, we have that

$$\mathcal{R}_M(e) = \{[u] \mid u \in \mathcal{G}_R(e)\} \quad \text{and} \quad \mathcal{G}_R(e) = \{\mathsf{nf}_R(u) \mid [u] \in \mathcal{R}_M(e)\}.$$

It follows that $\mathcal{R}_M(e_1) = \mathcal{R}_M(e_2)$ iff $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$. That is, $\mathcal{R}_M$ and $\mathcal{G}_R$ have the same equational theory. So, our investigations of completeness can be w.r.t. $\mathcal{G}_R$ instead of $\mathcal{R}_M$.

Lemma 18 ($\mathcal{G}$ and $\mathcal{C}$)**.** *For a confluent and terminating rewrite system R over Σ:*

1. $\mathcal{G}_R(e) = \mathscr{G}_R(\mathcal{R}(e))$, for an expression e.
2. $\mathcal{G}_R(e) = \mathcal{G}_R(e')$ iff $\mathcal{C}_R(e) = \mathcal{C}_R(e')$, for expressions e, e'.
3. $\mathcal{C}_R(e) = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e)))$, for an expression e.

Proof. Part (1) is shown by induction on the structure of e. For parts (2) and (3) we make use of Lemma 14. □

Definition 19 (**Well-Behaved Rewrite System).** Let R be a rewrite system over Σ. We say that R is *well-behaved* if it satisfies the following properties:

1. R consists of rules of the form $\ell \to r$ with $|r| \leq 1$ and $|\ell| > 1$.
2. R is confluent.
3. *Regularity*: For every x in $\Sigma \cup \{\varepsilon\}$, the R-ancestors $\mathsf{Ance}_R(x) = \{u \mid u \to_R^* x\}$ of x form a regular set $\mathcal{R}(e_x)$ for some regular expression e_x.
4. *Provability*: For every x in $\Sigma \cup \{\varepsilon\}$, the equation $e_x \equiv x$ is provable in KA_R. For $x = \varepsilon$ the equation instantiates to $e_\varepsilon \equiv 1$.

Condition (1) implies that every rule application strictly reduces the length of a string. That is, R is length-reducing and hence terminating. So, given Condition (1), we can require equivalently that R is locally confluent instead of having Condition (2). This result is known as Newman's Lemma (see [23, 12]). Recall that KA_R is the system of KA extended with equations corresponding to the rules of R.

Intuitively, the definition of well-behavedness for R enables two important algebraic constructions. First, the special form of the rules allows the automata-theoretic descendants construction (described in §2.3), which can be modeled in KA, because automata can be encoded as matrices. Then, the regularity requirement for the sets of R-ancestors implies that we can apply a homomorphism to obtain all the ancestors of a regular set. We can thus "close" a regular expression under the congruence induced by R.

Theorem 20 (**Completeness).** Let R be a well-behaved rewrite system over Σ. For all expressions e_1 and e_2, $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$ implies that $\mathsf{KA}_R \vdash e_1 \equiv e_2$.

Proof. Consider the following transformation steps on an arbitrary regular expression e:

1. *Descendants Construction*: As described in Theorem 15 we obtain an expression e' with $\mathsf{KA}_R \vdash e \equiv e'$ and $\mathcal{R}(e') = \mathsf{Desc}_R(\mathcal{R}(e))$.
2. *Ancestors Construction*: We describe below a transformation that gives us a new regular expression e'' with $\mathsf{KA}_R \vdash e' \equiv e''$ and $\mathcal{R}(e'') = \mathsf{Ance}_R(\mathcal{R}(e'))$.

We thus have $\mathsf{KA}_R \vdash e \equiv e''$ and $\mathcal{R}(e'') = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e)))$, which is equal to $\mathscr{C}_R(\mathcal{R}(e))$ by Lemma 14(4). It follows that $\mathcal{R}(e'') = \mathcal{C}_R(e)$ (see Definition 16).

Now, we apply the constructions (1) and (2) described above to the expressions e_1 and e_2 to obtain the expressions e_1'' and e_2'' with:

$$\mathsf{KA}_R \vdash e_1 \equiv e_1'' \qquad \mathcal{C}_R(e_1) = \mathcal{R}(e_1'') \qquad \mathsf{KA}_R \vdash e_2 \equiv e_2'' \qquad \mathcal{C}_R(e_2) = \mathcal{R}(e_2'')$$

From the hypothesis $\mathcal{G}_R(e_1) = \mathcal{G}_R(e_2)$ and Lemma 18(3) we get that $\mathcal{C}_R(e_1) = \mathcal{C}_R(e_2)$. It follows that $\mathcal{R}(e_1'') = \mathcal{R}(e_2'')$, and by completeness of KA for the standard interpretation $\mathcal{R}$ [70] we get that $\mathsf{KA} \vdash e_1'' \equiv e_2''$. Since we have proved in KA_R the equations

$$e_1 \equiv e_1'' \qquad\qquad e_1'' \equiv e_2'' \qquad\qquad e_2'' \equiv e_2$$

we conclude by transitivity that $\mathsf{KA}_R \vdash e_1 \equiv e_2$.

It remains to describe Step (2) of the above transformation to complete the proof. If u is an R-ancestor of a string v, then u is an R^{-1}-descendant of v (and conversely). Since R is well-behaved, the system R^{-1} only contains rules of the form $\varepsilon \to r$ and $a \to r$, where a is a letter. Moreover, for every x in $\Sigma \cup \{\varepsilon\}$ we have:

$$\mathsf{Ance}_R(x) = \mathsf{Desc}_{R^{-1}}(x) = \mathcal{R}(e_x)$$

for some expression e_x with $\mathsf{KA}_R \vdash e_x \equiv x$. Define the substitution θ as in Lemma 7. So,

$$\mathsf{Ance}_R(\mathcal{R}(e')) = \mathsf{Desc}_{R^{-1}}(\mathcal{R}(e')) = \mathcal{R}(\theta(e')).$$

We put $e'' = \theta(e')$. We have already shown that $\mathcal{R}(e'') = \mathsf{Ance}_R(\mathcal{R}(e'))$. It remains to see that $\mathsf{KA}_R \vdash e' \equiv e'' = \theta(e')$, which is implied by the provability assumption $\mathsf{KA}_R \vdash e_x \equiv x$, for every x in $\Sigma \cup \{\varepsilon\}$ (since R is well-behaved). $\square$

We will see now how the general completeness meta-theorem we have shown above (Theorem 20) can be used to obtain several concrete completeness results for the regular algebras of some simple finitely presented monoids.

Example 21 (Idempotence Hypotheses)**.** Consider the monoid $M = \langle a, b \mid aa \equiv a \rangle$. The rewrite system R contains only the rule $aa \to a$. In order to invoke Theorem 20 we have to verify that R is well-behaved (Definition 19):

- For the only rule $\ell = aa \to a = r$ of R, we have that $|r| = 1$ and $|\ell| > 1$.
- To show confluence of R, it is sufficient to show local confluence, since R is terminating. We have the following critical-pair lemma: Suppose that $u \to x$ and $u \to y$. If $x = y$, we are done. If $x \neq y$, then u, x, y must be of the following forms:

  $$u = v_1 a^{m+1} v_2 a^{n+1} v_3 \qquad x = v_1 a^m v_2 a^{n+1} v_3 \qquad y = v_1 a^{m+1} v_2 a^n v_3$$

  Notice now that $x, y \to v_1 a^m v_2 a^n v_3$, which establishes local confluence.
- For the R-ancestors of the letter a, we see that $\mathsf{Ance}_R(a) = \{a^n \mid n > 0\} = \mathcal{R}(e_a)$, where $e_a = a; a^*$. Reasoning in KA_R we show that $a \leq a; a^*$ and

  $$a; a^* \leq a \Longleftarrow a; a \leq a \Longleftarrow a; a \equiv a.$$

  We have thus shown that $\mathsf{KA}_R \vdash e_a \equiv a$.
- For ε and the letter b, we have $\mathsf{Ance}_R(b) = \{b\} = \mathcal{R}(b)$ and $\mathsf{Ance}_R(\varepsilon) = \{\varepsilon\} = \mathcal{R}(1)$.

Since the rewrite system R satisfies the conditions of Theorem 20, we get completeness of KA together with the equation $a; a \equiv a$ for the interpretations $\mathcal{G}_R$

and $\mathcal{R}_M$.

Example 22 (Self-Inverse Action)**.** We consider the monoid $M = \langle a, b \mid bb \equiv \varepsilon \rangle$ and the rewrite system R with the rule $bb \to \varepsilon$. We verify that R is well-behaved (Definition 19). It is easy to observe that the rules of R are of the appropriate form, and that R is confluent. For the regularity and provability assumption, we have $\mathsf{Ance}_R(\varepsilon) = \{(bb)^n \mid n \geq 0\} = \mathcal{R}(e_b)$, where $e_b = (b;b)^*$. Reasoning in KA_R we have $e_b \equiv (b;b)^* \equiv 1^* \equiv 1$. By Theorem 20, we get completeness of KA extended with $b;b \equiv 1$ for the interpretations $\mathcal{G}_R$ and $\mathcal{R}_M$.

2.5 Completeness: Partial Monoid Equations

We would like to generalize our result in a way that allows us to designate certain strings as being *non-well-formed* or *undefined*. Any string with a non-well-formed substring has to be discarded from the interpretation. For a string $a_1 \cdots a_k$ over the alphabet, we declare it to be non-well-formed using the equation $a_1 \cdots a_k \equiv \bot$, where $\bot$ is a special "undefined" symbol not in the alphabet.

We define a *partial monoid* to be an algebraic structure $(M, \cdot, 1_M, \bot_M)$ satisfying the monoid axioms, as well as the equations $x \cdot \bot_M = \bot_M$ and $\bot_M \cdot x = \bot_M$. The identity is 1_M, and $\bot_M$ is called the *undefined element* of M. In a presentation of a partial monoid

$$M_\bot = \langle \Sigma \mid x_1 \equiv y_1, x_2 \equiv y_2, \ldots, z_1 \equiv \bot, z_2 \equiv \bot, \ldots \rangle$$

we allow equations $x \equiv y$ between strings over Σ (call the collection of these R), as well as equations of the form $z \equiv \bot$, where z is a string over Σ ($\bot$ is not in Σ). Call $R_\bot$ the set of all equations of the presentation. In order to give a concrete description of the partial monoid, we consider the strings over the extended alphabet $\Sigma \cup \{\bot\}$. Let $\sim$ be the smallest congruence on $(\Sigma \cup \{\bot\})^*$ that contains $R_\bot$, as well as the axioms $x\bot \equiv \bot$ and $\bot x \equiv \bot$ for every x in $\Sigma \cup \{\bot\}$. The partial monoid $M_\bot$ is the monoid of strings $(\Sigma \cup \{\bot\})^*$ modulo the congruence $\sim$. The identity is the $\sim$-congruence class $[\varepsilon]$, and the undefined element is the class $[\bot]$.

Assumption 23. We collect a list of assumptions for (Σ, R, Z), where Σ is a finite alphabet, R is a rewrite system over Σ, and $Z \subseteq \Sigma^*$ is a nonempty set of "undefined" strings.

1. R is confluent and terminating.
2. ***Seamlessness property***: If xzy is a string with $z \in Z$, then every R-successor of xzy is of the form $x'z'y'$ with $z' \in Z$. In other words, if x has a substring in Z then every R-successor of x has a substring in Z.

Intuitively, seamlessness says that if a string contains a non-well-formed substring, then no R-rewriting can make it well-formed. So, R interacts well with undefinedness.

Note 24. Without loss of generality we can require that the undefined strings of Z are nonempty, i.e., not equal to ε. This is because if $\varepsilon \in Z$, then the monoids we define later become trivial (the identity is equal to the undefined element).

Example 25. Consider the alphabet $\Sigma = \{a, \bar{a}, b, \bar{b}\}$ and the rewrite system R with rules $aa \to a$ and $bb \to b$. The set $Z = \{a\bar{a}, b\bar{b}\}$ contains the undefined strings. Arguing as in Example 21 we see that R is terminating and confluent. For the seamlessness property, we consider the case of a substring $a\bar{a}$ (the case for $b\bar{b}$ is analogous). Every R-rewriting of $xa\bar{a}y$ gives a successor $x'a\bar{a}y'$, so we are done. Therefore, (Σ, R, Z) satisfies Assumption 23.

If we added the rule $ba \to b$ to R, then the seamlessness property would be violated, because $ba\bar{a} \to b\bar{a}$ by applying the newly introduced rule.

Definition 26 (Partial Coalesced Product)**.** Let (Σ, R, Z) satisfy Assumption 23. Let $I_R \subseteq \Sigma^*$ be the set R-irreducible strings and $J_R = I_R \setminus (\Sigma^* \cdot Z \cdot \Sigma^*)$. That is,

$$J_R = \{u \in \Sigma^* \mid u \text{ is } R\text{-irreducible and has no substring in } Z\}.$$

Define the (partial) *coalesced product* $\diamond$ on elements of J_R as follows:

$$u \diamond v = \begin{cases} \mathsf{nf}_R(uv), & \text{if } \mathsf{nf}_R(uv) \text{ has no substring in } Z; \\ \bot, & \text{if } \mathsf{nf}_R(uv) \text{ has a substring in } Z. \end{cases}$$

As defined, $\diamond$ is of type $J_R \times J_R \to J_R \cup \{\bot\}$. Extend it to a binary operation on $J_R \cup \{\bot\}$:

$$\bot \diamond \bot = \bot \qquad u \diamond \bot = \bot \qquad \bot \diamond u = \bot$$

for every $u \in J_R$. We lift the product $\diamond$ into a total operation on subsets of J_R:

$$A \diamond B = \{u \diamond v \mid u \diamond v \neq \bot, u \in A, v \in B\},$$

where $A, B \subseteq J_R$. The structure $(J_R \cup \{\bot\}, \diamond, \varepsilon, \bot)$ is said to be the partial monoid (see Lemma 27 below) that corresponds to the triple (Σ, R, Z).

Lemma 27 (Partial Monoid From Rewrite System)**.** Let (Σ, R, Z) satisfy Assumption 23. Then, the structure $(J_R \cup \{\bot\}, \diamond, \varepsilon, \bot)$ is a partial monoid and is isomorphic to $\langle \Sigma \mid R_\bot \rangle$, where $R_\bot$ contains equations for the rules of R and an equation $z \equiv \bot$ for every $z \in Z$.

Proof. The proof that $J_R \cup \{\bot\}$ is a partial monoid relies on properties of normal forms. Now, we define the map $h : (\Sigma \cup \{\bot\})^* \to J_R \cup \{\bot\}$ as follows:

$$h(x) = \begin{cases} \bot, & \text{if } x \text{ contains an occurrence of } \bot \\ \bot, & \text{if } x \in \Sigma^* \text{ and } \mathsf{nf}_R(x) \text{ has a substring in } Z \\ \mathsf{nf}_R(x), & \text{if } x \in \Sigma^* \text{ and } \mathsf{nf}_R(x) \text{ has no substring in } Z \end{cases}$$

Claim 28. For all strings x, y over $\Sigma \cup \{\bot\}$, it holds that $h(xy) = h(x) \diamond h(y)$.

Recall that $\sim$ is the smallest relation that contains the pairs (u, v) of R, the pairs $(z, \bot)$ for all $z \in Z$, the partial monoid axioms $(x\bot, \bot)$ and $(\bot x, \bot)$ for $x \in \Sigma \cup \{\bot\}$, and is closed under the rules of equational logic (reflexivity, symmetry, transitivity, congruence).

Claim 29. For every pair (x, y) of $\sim$ we have that $h(x) = h(y)$.

The above claim asserts that h has the same value on members of an equivalence class of $\sim$. So, we define the map $\hat{h} : (\Sigma \cup \{\bot\})^* / {\sim} \to J_R \cup \{\bot\}$ by $[x] \mapsto h(x)$. We also claim that $\hat{h}$ is a partial monoid homomorphism from $\langle \Sigma \mid R_\bot \rangle$ to $J_R \cup \{\bot\}$. Indeed, we have:

$$\hat{h}([x] \cdot [y]) = \hat{h}([xy]) = h(xy) = h(x) \diamond h(y) = \hat{h}([x]) \diamond \hat{h}([y]),$$

as well as $\hat{h}([\varepsilon]) = h(\varepsilon) = \varepsilon$ and $\hat{h}([\bot]) = h(\bot) = \bot$. That is, $\hat{h}$ commutes with the partial monoid operations. Clearly, $\hat{h}$ is surjective. Finally, we claim that it is injective. Consider strings x, y over $\Sigma \cup \{\bot\}$ with $\hat{h}([x]) = \hat{h}([y])$, that is, $h(x) = h(y)$.

- If $h(x) = h(y) = \bot$ then $x \sim \bot$ and $y \sim \bot$, hence $x \sim y$ and $[x] = [y]$.
- If $h(x), h(y) \neq \bot$ then $\mathsf{nf}_R(x) = \mathsf{nf}_R(y)$, hence $x \sim y$ and $[x] = [y]$.

We have thus established that $\hat{h}$ is an isomorphism. □

Definition 30 (Language Interpretations)**.** Suppose that (Σ, R, Z) satisfies Assumption 23. Define $[\bot]_\Sigma = \mathsf{Ance}_R(\Sigma^* \cdot Z \cdot \Sigma^*)$ and for a string u over Σ:

$$[u]_\Sigma = \begin{cases} \mathsf{Ance}_R(\mathsf{nf}_R(u)), & \text{if } \mathsf{nf}_R(u) \text{ has no substring in } Z \\ [\bot]_\Sigma, & \text{if } \mathsf{nf}_R(u) \text{ has a substring in } Z \end{cases}$$

For a language $L \subseteq \Sigma^*$ we define $\mathscr{G}_\perp(L) \subseteq J_R$ and $\mathscr{C}_\perp(L) \subseteq \Sigma^*$ as follows:

$$\mathscr{G}_\perp(L) = \{\mathsf{nf}_R(u) \mid u \in L\} \setminus [\perp]_\Sigma \qquad \mathscr{C}_\perp(L) = [\perp]_\Sigma \cup \bigcup_{u \in L} [u]_\Sigma$$

Now, the interpretation $\mathcal{G}_\perp$ sends a regular expression to a subset of J_R:

$$\mathcal{G}_\perp(a) = \mathscr{G}_\perp(\{a\}) \qquad \mathcal{G}_\perp(e_1 + e_2) = \mathcal{G}_\perp(e_1) \cup \mathcal{G}_\perp(e_2)$$

$$\mathcal{G}_\perp(0) = \emptyset \qquad \mathcal{G}_\perp(e_1; e_2) = \mathcal{G}_\perp(e_1) \diamond \mathcal{G}_\perp(e_2)$$

$$\mathcal{G}_\perp(1) = \{\varepsilon\} \qquad \mathcal{G}_\perp(e^*) = \bigcup_{n \geq 0} \mathcal{G}_\perp(e)^{\langle n \rangle}$$

where $A^{\langle 0 \rangle} = \mathcal{G}_\perp(1)$ and $A^{\langle n+1 \rangle} = A^{\langle n \rangle} \diamond A$. Define $\mathcal{C}_\perp(e) = \mathscr{C}_\perp(\mathcal{R}(e))$. The interpretation $\mathcal{G}_\perp$ discards the undefined strings, but $\mathcal{C}_\perp$ adds them all in.

Lemma 31. Let (Σ, R, Z) satisfy Assumption 23. For all languages $L, L' \subseteq \Sigma^*$:

1. $\mathscr{C}_\perp(L) = \mathscr{C}_\perp(\mathscr{G}_\perp(L))$.
2. $\mathscr{G}_\perp(L) = \mathscr{G}_\perp(\mathscr{C}_\perp(L))$.
3. $\mathscr{G}_\perp(L) = \mathscr{G}_\perp(L')$ iff $\mathscr{C}_\perp(L) = \mathscr{C}_\perp(L')$.
4. $\mathscr{C}_\perp(L) = \mathsf{Ance}_R(\mathsf{Desc}_R(L)) \cup [\perp]_\Sigma$.

The above are the analog of Lemma 14. As an analog of Lemma 18, we have:

5. $\mathcal{G}_\perp(e) = \mathscr{G}_\perp(\mathcal{R}(e))$, for an expression e.
6. $\mathcal{G}_\perp(e) = \mathcal{G}_\perp(e')$ iff $\mathcal{C}_\perp(e) = \mathcal{C}_\perp(e')$, for expressions e, e'.
7. $\mathcal{C}_\perp(e) = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e))) \cup [\perp]_\Sigma$, for an expression e.

Proof. Similar to the proofs of Lemma 14 and Lemma 18. □

Definition 32 (Well-Behaved). Let (Σ, R, Z) satisfy Assumption 23. We say that it is *well-behaved* if it satisfies additionally the following properties:

1. R consists of rules of the form $\ell \to r$ with $|r| \leq 1$ and $|\ell| > 1$.
2. *Regularity*: For every x in $\Sigma \cup \{\varepsilon\}$, the R-ancestors $\mathsf{Ance}_R(x) = \{u \mid u \to_R^* x\}$ of x form a regular set $\mathcal{R}(e_x)$ for some regular expression e_x.

3. *Regular undefined set*: There is a regular expression e_Z with $\mathcal{R}(e_Z) = Z$.
4. *Provability*: For every x in $\Sigma \cup \{\varepsilon\}$, the equation $e_x \equiv x$ is provable in KA_R.

We will write $\mathsf{KA}_\perp$ for the system KA_R extended with the equation $e_Z \equiv 0$. Notice that if (Σ, R, Z) is well-behaved, then so is R (in the sense of Definition 19).

Lemma 33 (Undefined Class). Let (Σ, R, Z) be well-behaved. The set $[\perp]_\Sigma$ of undefined strings is regular. For the corresponding expression $e_\perp$ is holds that $\mathsf{KA}_\perp \vdash e_\perp \equiv 0$.

Proof. Recall that $[\perp]_\Sigma = \mathsf{Ance}_R(\Sigma^* \cdot Z \cdot \Sigma^*)$ from Definition 30. Part of the definition of well-behavedness is that Z is regular, where $\mathcal{R}(e_Z) = Z$ for some expression e_Z. It follows that the set $\Sigma^* \cdot Z \cdot \Sigma^*$ is also regular. The corresponding expression is $e = e_U; e_Z; e_U$, where $e_U = (\sum_a a)^*$ is the universal expression with $\mathcal{R}(e_U) = \Sigma^*$. Since the rewrite system R is well-behaved, the inverse system R^{-1} satisfies the conditions of Lemma 7. Let θ be the substitution defined in the lemma. We then have

$$[\perp]_\Sigma = \mathsf{Ance}_R(\mathcal{R}(e)) = \mathcal{R}(\theta(e)).$$

Since R is well-behaved, we get from the provability condition that $\mathsf{KA}_R \vdash e \equiv \theta(e)$. Finally, we put $e_\perp = \theta(e)$. Reasoning in $\mathsf{KA}_\perp$, we prove the equations:

$$e_\perp \equiv \theta(e) \equiv e \equiv e_U; e_Z; e_U \equiv e_U; 0; e_U \equiv 0.$$

We have thus established $\mathsf{KA}_\perp \vdash e_\perp \equiv 0$, and the proof is complete. □

Theorem 34 (Completeness). Suppose that (Σ, R, Z) is well-behaved. For all expressions e_1 and e_2, $\mathcal{G}_\perp(e_1) = \mathcal{G}_\perp(e_2)$ implies that $\mathsf{KA}_\perp \vdash e_1 \equiv e_2$.

Proof. Consider the following transformation steps on an arbitrary regular expression e:

1. *Descendants Construction*: As described in Theorem 15, we get an expression e' with $\mathsf{KA}_R \vdash e \equiv e'$ and $\mathcal{R}(e') = \mathsf{Desc}_R(\mathcal{R}(e))$.
2. *Ancestors Construction*: Define the substitution θ by $\varepsilon \mapsto e_\varepsilon$ and $a \mapsto e_a$ for every letter $a \in \Sigma$. As in Theorem 20: $\mathsf{KA}_R \vdash e' \equiv \theta(e')$ and $\mathcal{R}(\theta(e')) = \mathsf{Ance}_R(\mathcal{R}(e'))$.
3. *Undefined Class*: It was shown in Lemma 33 that there is an expression $e_\perp$ such that $\mathsf{KA}_\perp \vdash e_\perp \equiv 0$ and $\mathcal{R}(e_\perp) = [\perp]_\Sigma$. We put $\hat{e} = \theta(e') + e_\perp$.

Combining the above steps we get $\mathsf{KA}_\perp \vdash e \equiv e' \equiv \theta(e') \equiv \theta(e') + e_\perp = \hat{e}$, and

$$\mathcal{R}(\hat{e}) = \mathcal{R}(\theta(e')) \cup \mathcal{R}(e_\perp) = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e))) \cup [\perp]_\Sigma,$$

which is equal to $\mathcal{C}_\perp(e)$ using Lemma 31(7). We have thus shown that $\mathcal{R}(\hat{e}) = \mathcal{C}_\perp(e)$. We apply this construction to the expressions e_1 and e_2 to obtain $\hat{e}_1$ and $\hat{e}_2$ with:

$$\mathsf{KA}_\perp \vdash e_1 \equiv \hat{e}_1 \qquad \mathcal{C}_\perp(e_1) = \mathcal{R}(\hat{e}_1) \qquad \mathsf{KA}_\perp \vdash e_2 \equiv \hat{e}_2 \qquad \mathcal{C}_\perp(e_2) = \mathcal{R}(\hat{e}_2)$$

The hypothesis $\mathcal{G}_\perp(e_1) = \mathcal{G}_\perp(e_2)$ and Lemma 31(6) imply $\mathcal{C}_\perp(e_1) = \mathcal{C}_\perp(e_2)$. It follows that $\mathcal{R}(\hat{e}_1) = \mathcal{R}(\hat{e}_2)$ and by completeness of KA for the interpretation $\mathcal{R}$ we get that $\mathsf{KA} \vdash \hat{e}_1 \equiv \hat{e}_2$. We conclude that $\mathsf{KA}_\perp \vdash e_1 \equiv e_2$. □

2.6 Completeness: Typed Monoid Equations

We further generalize the partial monoid setting of §2.5 by assuming even more structure on the strings and the rewrite system. One major difference from the partial monoid case is the introduction of a new category of primitive symbols, the *subidentities*, which allow the encoding of Booleans. Using this general typed framework, we will show later how to cover several examples, among which: plain KAT, KAT with simple Hoare hypotheses $b; p; c \equiv 0$, KAT with hypotheses

$c; p \equiv c$, and NetKAT. These examples attest to the generality and wide applicability of our technique.

Assumption 35. We collect a list of assumptions for (P, Id, R, Z). The sets P and Id are finite alphabets which we call the *action symbols* and *subidentities* respectively. We write $p, q, r, \ldots$ to vary over actions symbols, and $\alpha, \beta, \gamma, \ldots$ to vary over subidentities. We put

$$\Sigma = Id \cup \{\alpha p \beta \mid p \in P \text{ and } \alpha, \beta \in Id\},$$

and we call Σ the *aggregated alphabet*. We write $a, b, c, \ldots$ to vary over arbitrary letters of Σ. Examples of nonempty strings over Σ are: α, $\alpha\,\alpha$, $\alpha p \beta$, $\alpha\,\beta$, $\alpha\,\alpha p \beta$, $\alpha\,\beta p \gamma\,\delta$, $\alpha p \beta\,\beta q \gamma\,\gamma$, and so on. Σ^+ denotes the set of nonempty strings over Σ.

Let R be a string rewrite system over Σ, and $Z \subseteq \Sigma^+$ be a nonempty set of nonempty strings (the "undefined" ones). We require the following:

1. The rules of R involve *nonempty* strings. R includes at least the rules:

   $$\alpha\,\alpha \to \alpha \qquad\qquad \alpha\,\alpha p \beta \to \alpha p \beta \qquad\qquad \alpha p \beta\,\beta \to \alpha p \beta$$

   for all subidentites $\alpha, \beta \in Id$ and every action symbol $p \in P$.
2. R is confluent and terminating.
3. The set Z of undefined strings contains at least the following:

   $$\alpha\,\beta\ \ (\alpha \neq \beta) \quad\ \alpha\,\beta p \gamma\ \ (\alpha \neq \beta) \quad\ \alpha p \beta\,\gamma\ \ (\beta \neq \gamma) \quad\ \alpha p \beta\,\gamma q \delta\ \ (\beta \neq \gamma)$$
4. ***Seamlessness property***: If $x \in \Sigma^+$ has an undefined substring (in Z), then every R-successor of x also has an undefined substring.

As in Assumption 23, Condition (4) says that R preserves non-well-definedness.

An immediate consequence of the seamlessness property (described above) is the following ***closure property for*** Z: If the subidentity α is in Z, then $\alpha p \beta$ and

$\beta p\alpha$ are also in Z for every subidentity β and every action symbol p.

Example 36 (Kleene algebra with tests)**.** Let P be a finite set of actions symbols, and Id be a disjoint finite set of *atoms*. Suppose that R contains only the required rules, and Z contains only the required undefined strings. For confluence of R, we simply observe that all critical pairs are trivial. For the seamlessness property we examine all the cases where an undefined string is part of a redex:

$$
\begin{array}{lll}
\text{for } \alpha \neq \beta : & \alpha\,\underline{\alpha\,\beta} \to_R \underline{\alpha\,\beta} & \underline{\alpha\,\beta}\,\beta \to_R \underline{\alpha\,\beta} \\
\text{for } \beta \neq \gamma : & \alpha p\beta\,\underline{\beta\,\gamma} \to_R \underline{\alpha p\beta\,\gamma} & \underline{\beta\,\gamma}\,\gamma p\delta \to_R \underline{\beta\,\gamma p\delta} \\
\text{for } \alpha \neq \beta : & \alpha\,\underline{\alpha\,\beta p\gamma} \to_R \underline{\alpha\,\beta p\gamma} & \underline{\alpha\,\beta p\gamma}\,\gamma \to_R \underline{\alpha\,\beta p\gamma} \\
\text{for } \beta \neq \gamma : & \alpha p\beta\,\underline{\beta\,\gamma q\delta} \to_R \underline{\alpha p\beta\,\gamma q\delta} & \\
\text{for } \beta \neq \gamma : & \alpha\,\underline{\alpha p\beta\,\gamma p\delta} \to_R \underline{\alpha p\beta\,\gamma p\delta} & \underline{\alpha p\beta\,\gamma p\delta}\,\delta \to_R \underline{\alpha p\beta\,\gamma p\delta}
\end{array}
$$

The underlined parts in the reductions above are the undefined substrings. The cases for $\alpha p\beta\,\gamma$ are analogous to the ones for $\alpha\,\beta p\gamma$. Generally, no application of a rule of R can eliminate a part $\ldots\alpha\,\beta\ldots$ with $\alpha \neq \beta$. So, the quadruple (P, Id, R, Z) satisfies Assumption 35.

Definition 37 (Typed Coalesced Product)**.** Let (P, Id, R, Z) satisfy Assumption 35. Let $I_R \subseteq \Sigma^+$ be the R-irreducible strings of Σ^+ and $J_R = I_R \backslash (\Sigma^* \cdot Z \cdot \Sigma^*)$. That is,

$$J_R = \{u \in \Sigma^+ \mid u \text{ is } R\text{-irreducible and has no substring in } Z\}.$$

Define the *(untyped) coalesced product* $\diamond$ as in Definition 26. For x in J_R, we write $x : \alpha \to \beta$ to mean that x starts with α and ends with β. The expression $\alpha \to \beta$ is said to be the *type* of x. Observe that $\alpha : \alpha \to \alpha$ for every subidentity α and $\alpha p\beta : \alpha \to \beta$ for the composite letters of Σ. We introduce a family of *undefined* symbols $\perp_{\alpha\beta} : \alpha \to \beta$ for all subidentities α, β. We define the *typed coalesced* *product* $\diamond$ as:

$$x \diamond y = \begin{cases} \mathsf{nf}_R(xy), & \text{if } \mathsf{nf}_R(xy) \text{ has no substring in } Z \\ \bot_{\alpha\gamma}, & \text{if } \mathsf{nf}_R(xy) \text{ has a substring in } Z \end{cases}$$

for $x : \alpha \to \beta$ and $y : \beta \to \gamma$ in J_R. Extend the operation to account for undefined operands:

$$\frac{x : \alpha \to \beta \text{ in } J_R}{x \diamond \bot_{\beta\gamma} = \bot_{\alpha\gamma}} \qquad \frac{y : \beta \to \gamma \text{ in } J_R}{\bot_{\alpha\beta} \diamond y = \bot_{\alpha\gamma}} \qquad \bot_{\alpha\beta} \diamond \bot_{\beta\gamma} = \bot_{\alpha\beta}$$

Observe that $\diamond$ can also be seen as a family $(\diamond_{\alpha\beta\gamma})_{\alpha,\beta,\gamma\in Id}$ of operations indexed by types.

Observation 38 (Typed Partial Monoid)**.** Let (P, Id, R, Z) satisfy Assumption 35. Notice that this quadruple gives rise to a typed monoid (i.e., a small category):

– The *objects* (types) are the subidentities Id.

– The elements of type $\alpha \to \beta$ are $\mathrm{Hom}(\alpha, \beta) = \{x \in J_R \mid x : \alpha \to \beta\} \cup \{\bot_{\alpha\beta}\}$.

– The typed coalesced product $\diamond$ is the composition operation of the category:

$$\frac{x \text{ in } \mathrm{Hom}(\alpha,\beta) \qquad y \text{ in } \mathrm{Hom}(\beta,\gamma) \qquad z \text{ in } \mathrm{Hom}(\gamma,\delta)}{(x \diamond y) \diamond z = x \diamond (y \diamond z)}$$

The associativity of $\diamond$ relies on the confluence of R and the seamlessness property.

– For every subidentity α, we have an *identity* element $\alpha \in \mathrm{Hom}(\alpha, \alpha)$ of the category:

$$\frac{x \text{ in } \mathrm{Hom}(\alpha,\beta)}{\alpha \diamond x = x} \qquad \frac{x \text{ in } \mathrm{Hom}(\alpha,\beta)}{x \diamond \beta = x}$$

For the first equation above, we examine the cases where x is equal to α or of the form $\alpha p \alpha' x'$. Since R contains the rule $\alpha\,\alpha \to \alpha$, we have that $\alpha \diamond \alpha = \mathsf{nf}_R(\alpha\,\alpha) = \mathsf{nf}_R(\alpha) = \alpha$. Similarly, $\alpha \diamond \alpha p \alpha' x' = \mathsf{nf}_R(\alpha\,\alpha p \alpha' x') = \mathsf{nf}_R(\alpha p \alpha' x') = \alpha p \alpha' x'$, since $\alpha\,\alpha p \alpha' \to_R \alpha p \alpha'$.

– Every hom-set $\mathrm{Hom}(\alpha, \beta)$ contains a distinguished *undefined* element $\bot_{\alpha\beta} : \alpha \to \beta$.

$$\frac{x \text{ in } \mathrm{Hom}(\alpha, \beta)}{x \diamond \bot_{\beta\gamma} = \bot_{\alpha\gamma}} \qquad\qquad \frac{x \text{ in } \mathrm{Hom}(\beta, \gamma)}{\bot_{\alpha\beta} \diamond x = \bot_{\alpha\gamma}}$$

The above constants and operations constitute a *typed partial monoid*, i.e., a small category with a distinguished undefined element for every homset.

Definition 39 (Language Interpretations)**.** Let (P, Id, R, Z) satisfy Assumption 35. Define $[\bot]_\Sigma = \mathsf{Ance}_R(\Sigma^* \cdot Z \cdot \Sigma^*)$, and for a string $u \in \Sigma^+$ we put:

$$[u]_\Sigma = \begin{cases} \mathsf{Ance}_R(\mathsf{nf}_R(u)), & \text{if } \mathsf{nf}_R(u) \text{ has no substring in } Z \\ [\bot]_\Sigma, & \text{if } \mathsf{nf}_R(u) \text{ has a substring in } Z \end{cases}$$

For a language $L \subseteq \Sigma^+$ we define $\mathscr{G}_\bot(L) \subseteq J_R$ and $\mathscr{C}_\bot(L) \subseteq S$ as follows:

$$\mathscr{G}_\bot(L) = \{\mathsf{nf}_R(u) \mid u \in L\} \setminus [\bot]_\Sigma \qquad\qquad \mathscr{C}_\bot(L) = [\bot]_\Sigma \cup \textstyle\bigcup_{u \in L} [u]_\Sigma$$

Now, the interpretation $\mathcal{G}_\bot$ sends a regular expression over Σ to a subset of J_R:

$$\begin{aligned} \mathcal{G}_\bot(a) &= \{\mathsf{nf}_R(a)\} \setminus [\bot]_\Sigma & \mathcal{G}_\bot(e_1 + e_2) &= \mathcal{G}_\bot(e_1) \cup \mathcal{G}_\bot(e_2) \\ \mathcal{G}_\bot(0) &= \emptyset & \mathcal{G}_\bot(e_1; e_2) &= \mathcal{G}_\bot(e_1) \diamond \mathcal{G}_\bot(e_2) \\ \mathcal{G}_\bot(1) &= \mathscr{G}_\bot(Id) & \mathcal{G}_\bot(e^*) &= \textstyle\bigcup_{n \geq 0} \mathcal{G}_\bot(e)^{\langle n \rangle} \end{aligned}$$

We define $\mathcal{C}_\bot(e) = \mathscr{C}_\bot(\mathcal{R}(e))$ for an expression e with $\mathcal{R}(e) \subseteq \Sigma^+$. As in Definition 30, the interpretation $\mathcal{G}_\bot$ discards the undefined strings, but $\mathcal{C}_\bot$ adds them all in.

Lemma 40. Let (P, Id, R, Z) satisfy Assumption 35. For $L, L' \subseteq \Sigma^+$ we have:

1. $\mathscr{C}_\bot(L) = \mathscr{C}_\bot(\mathscr{G}_\bot(L))$.
2. $\mathscr{G}_\bot(L) = \mathscr{G}_\bot(\mathscr{C}_\bot(L))$.
3. $\mathscr{G}_\bot(L) = \mathscr{G}_\bot(L')$ iff $\mathscr{C}_\bot(L) = \mathscr{C}_\bot(L')$.
4. $\mathscr{C}_\bot(L) = \mathsf{Ance}_R(\mathsf{Desc}_R(L)) \cup [\bot]_S$.

The above are analogous to Lemmas 14 and 31. For all expressions e, e' we have:

5. $\mathcal{G}_\perp(e) = \bigcup_{x \in \mathcal{R}(e)} \mathcal{G}_\perp(x)$.
6. $\mathcal{G}_\perp(e) = \mathscr{G}_\perp(\mathcal{R}(e))$.
7. $\mathcal{G}_\perp(e) = \mathcal{G}_\perp(e')$ iff $\mathcal{C}_\perp(e) = \mathcal{C}_\perp(e')$.
8. $\mathcal{C}_\perp(e) = \mathsf{Ance}_R(\mathsf{Desc}_R(\mathcal{R}(e))) \cup [\perp]_S$.

For the parts (6), (7), and (8) we have the implicit assumption that $\mathcal{R}(e), \mathcal{R}(e') \subseteq \Sigma^+$. For a string $a_1 a_2 \cdots a_n$, we write $\mathcal{G}_\perp(a_1 a_2 \dots a_n)$ to mean $\mathcal{G}_\perp(a_1; a_2; \cdots; a_n)$. Moreover, $\mathcal{G}_\perp(\varepsilon)$ is notation for $\mathcal{G}_\perp(1)$.

Proof. Similar to the proof of Lemma 31. □

Definition 41 (Well-Behaved). Let (P, Id, R, Z) satisfy Assumption 35. We say that it is *well-behaved* if it satisfies additionally the following properties:

1. R consists of rules of the form $\ell \to r$ with $|r| = 1$ and $|\ell| > 1$.
2. *Regularity*: For every letter a in Σ, the R-ancestors $\mathsf{Ance}_R(a) = \{u \mid u \to_R^* a\}$ of a form a regular set $\mathcal{R}(e_a)$ for some regular expressions e_a.
3. *Regular undefined set*: There is a regular expression e_Z with $\mathcal{R}(e_Z) = Z$.
4. *Provability*: For every letter a in Σ, the equation $e_a \equiv a$ is provable in KA_R.

$\mathsf{KA}_{\mathit{Id}}$ denotes the extension of KA_R with the equations $e_Z \equiv 0$ and $\sum_{\alpha \in \mathit{Id}} \alpha \equiv 1$.

Example 42 (Kleene algebra with tests)**.** Consider the quadruple (P, Id, R, Z) of Example 36, which satisfies Assumption 35. We claim that it is, in fact, well-behaved. The rules are of the right form. For the regularity condition, we have:

$$\mathsf{Ance}_R(\alpha) = \mathcal{R}(e_\alpha), \text{ where } e_\alpha = \alpha; \alpha^*$$
$$\mathsf{Ance}_R(\alpha p \beta) = \mathcal{R}(e_{\alpha p \beta}), \text{ where } e_{\alpha p \beta} = \alpha^*; \alpha p \beta; \beta^*$$

The set Z of undefined strings is finite and hence regular. We put

$$e_Z = \textstyle\sum_{\alpha \neq \beta} \alpha;\beta + \sum_{\alpha \neq \beta, \gamma} \alpha;\beta p \gamma + \sum_{\alpha, \beta \neq \gamma} \alpha p \beta;\gamma + \sum_{\alpha, \beta \neq \gamma, \delta} \alpha p \beta;\gamma q \delta.$$

For the provability condition, we first show $\mathsf{KA}_R \vdash \alpha \equiv \alpha;\alpha^*$ as in Example 21. Reasoning in KA_R, we also have that

$$e_{\alpha p \beta} \equiv \alpha^*;\alpha p \beta;\beta^* \equiv \alpha^*;\alpha;\alpha p \beta;\beta;\beta^* \equiv \alpha;\alpha p \beta;\beta \equiv \alpha p \beta.$$

We have used above equations of R, as well as the proved $\alpha \equiv e_\alpha$ and $\beta \equiv e_\beta$.

Theorem 43 (Completeness). Suppose that the quadruple (P, Id, R, Z) is well-behaved. For all expressions e_1 and e_2 over Σ, $\mathcal{G}_\perp(e_1) = \mathcal{G}_\perp(e_2)$ implies that $\mathsf{KA}_{\mathit{Id}} \vdash e_1 \equiv e_2$.

Proof. The proof is very similar to the one for plain partiality (see Theorem 34). The only noteworthy difference is that we need to account for the case where $\mathcal{R}(e_1)$ or $\mathcal{R}(e_2)$ contains the empty string, in order to be able to invoke Lemma 40. So, we need to transform an arbitrary regular expression e into e' with $\mathsf{KA}_{\mathit{Id}} \vdash e \equiv e'$ and $\mathcal{R}(e') \subseteq \Sigma^+$. We put $e' = (\sum_{\alpha \in \mathit{Id}} \alpha);e$ and we see immediately that

$$\mathsf{KA}_{\mathit{Id}} \vdash e \equiv 1;e \equiv (\textstyle\sum_{\alpha \in \mathit{Id}} \alpha);e \equiv e'.$$

For an arbitrary language $L \subseteq J_R$ we have that $\mathit{Id} \diamond L = L$, and therefore $\mathcal{G}_\perp(e') = \mathcal{G}_\perp(e)$. By virtue of these observations, we can assume without loss of generality that $\mathcal{R}(e_1)$ and $\mathcal{R}(e_2)$ are contained in Σ^+. The proof then proceeds exactly as in Theorem 34. $\square$

2.7 Applications

Theorem 43 gives us four completeness results as corollaries. First, we show that KAT is complete for the standard interpretation of KAT terms as sets of guarded strings. We then consider the case of KAT extended with simple Hoare hypotheses $b; p; c \equiv 0$ (tests b, c, atomic action p), and with hypotheses $c; p \equiv c$. We conclude with a completeness proof for NetKAT.

2.7.1 Kleene algebra with tests

We consider an alternative presentation of Kleene algebra with tests (KAT) in *reduced* form. The primitive symbols are either actions $p, q, r, \ldots$ in P or atoms $\alpha, \beta, \gamma, \ldots$ in Id. *Reduced KAT* extends KA with the extra axioms:

$$\alpha; \alpha \equiv \alpha \qquad \alpha; \beta \equiv 0, \text{ if } \alpha \neq \beta \qquad \textstyle\sum_{\alpha \in Id} \alpha \equiv 1$$

We want to give now another equivalent presentation of KAT where the primitive symbols are from $\Sigma = Id \cup \{\alpha p \beta \mid \alpha, \beta \in Id\}$. We propose the axiomatization for *KAT over* Σ:

$$\begin{array}{lll}
\alpha; \alpha \equiv \alpha & \alpha; \beta \equiv 0, \text{ if } \alpha \neq \beta & \alpha p \beta; \gamma q \delta \equiv 0, \text{ if } \beta \neq \gamma \\
\alpha; \alpha p \beta \equiv \alpha p \beta & \alpha; \beta p \gamma \equiv 0, \text{ if } \alpha \neq \beta & \sum_{\alpha \in Id} \alpha \equiv 1 \\
\alpha p \beta; \beta \equiv \alpha p \beta & \alpha p \beta; \gamma \equiv 0, \text{ if } \beta \neq \gamma &
\end{array}$$

Notice that this axiomatization is essentially the same as KA_{Id} for the quadruple (P, Id, R, Z) which we considered in Examples 36 and 42.

Theorem 44. Let $\mathcal{G}_{\mathsf{KAT}}$ be the standard interpretation of Reduced KAT expressions as sets of guarded strings [82]. Then, $\mathcal{G}_{\mathsf{KAT}}(e_1) = \mathcal{G}_{\mathsf{KAT}}(e_2)$ implies $e_1 \equiv e_2$ in Reduced KAT.

Proof. Let (P, Id, R, Z) be the rewrite system of Example 36, which is well-behaved (proved in Example 42). Now, we define the substitutions θ and θ^{-1} as follows:

$$\theta : p \mapsto \textstyle\sum_{\alpha,\beta} \alpha p \beta \qquad\qquad \theta^{-1} : \alpha p \beta \mapsto \alpha; p; \beta$$

The substitution θ translates from the language of Reduced KAT to the language of KAT over Σ, and θ^{-1} goes the other way. Let h be the function that transforms a guarded string to the corresponding R-irreducible string of J_R:

$$h : \alpha_0 p_1 \alpha_1 p_2 \alpha_2 \cdots \alpha_{n-1} p_n \alpha_n \mapsto \alpha_0 \, \alpha_0 p_1 \alpha_1 \, \alpha_1 p_2 \alpha_2 \, \cdots \alpha_{n-1} p_n \alpha_n$$

For an expression e over Σ, we have $\mathcal{G}_\perp(e) = h(\mathcal{G}_{\mathsf{KAT}}(\theta^{-1}(e)))$. Reasoning in Reduced KAT:

$$p \equiv 1; p; 1 \equiv (\textstyle\sum_{\alpha \in Id} \alpha); p; (\sum_{\beta \in At} \beta) \equiv \sum_{\alpha,\beta \in Id} \alpha; p; \beta.$$

It follows that we can show $e \equiv \theta^{-1}(\theta(e))$ in Reduced KAT. By soundness of Reduced KAT for the interpretation $\mathcal{G}_{\mathsf{KAT}}$ we also obtain that $\mathcal{G}_{\mathsf{KAT}}(e) = \mathcal{G}_{\mathsf{KAT}}(\theta^{-1}(\theta(e)))$. So,

$$\begin{aligned}
\mathcal{G}_{\mathsf{KAT}}(e_1) = \mathcal{G}_{\mathsf{KAT}}(e_2) &\implies \mathcal{G}_{\mathsf{KAT}}(\theta^{-1}(\theta(e_1))) = \mathcal{G}_{\mathsf{KAT}}(\theta^{-1}(\theta(e_2))) \\
&\implies h(\mathcal{G}_{\mathsf{KAT}}(\theta^{-1}(\theta(e_1)))) = h(\mathcal{G}_{\mathsf{KAT}}(\theta^{-1}(\theta(e_2)))) \\
&\implies \mathcal{G}_\perp(\theta(e_1)) = \mathcal{G}_\perp(\theta(e_2)) \\
&\implies \mathsf{KAT} \vdash \theta(e_1) = \theta(e_2).
\end{aligned}$$

The last implication is by Theorem 43. Since the (translations of) all the axioms of KAT over Σ are provable in Reduced KAT, we conclude that $e_1 \equiv e_2$ is also provable. $\square$

2.7.2 KAT with simple Hoare hypotheses

A *simple Hoare assertion* is an expression of the form $\{b\}p\{c\}$, where b, c are tests and p is an atomic action. It can be encoded in KAT with any one of the following equations

$$b; p; \neg c \equiv 0 \qquad\qquad b; p \equiv b; p; c \qquad\qquad b; p \leq p; c$$

The equation $b; p; \neg c \equiv 0$ is equivalent to the conjunction of the equations $\beta; p; \gamma \equiv 0$, where β, γ are atoms with $\beta \leq b$ and $\gamma \leq \neg c$. So, without loss of generality we restrict attention to assertions of the form $\beta; p; \gamma \equiv 0$, where β, γ are atoms and p is an atomic action.

Let Z_h be a finite collection of strings of the form $\gamma p \delta$, where γ, δ are atoms and p is an action symbol, and H be the set of equations $\gamma; p; \delta \equiv 0$ for every $\gamma p \delta$ in Z_h. We write $\mathsf{KAT}{+}H$ for the extension of KAT with extra axioms H. We also define the interpretation $\mathcal{G}_h$ by $\mathcal{G}_h(e) = \mathcal{G}_{\mathsf{KAT}}(e) \setminus W$, where W is the set of strings containing some $\gamma p \delta$ in Z_h.

Theorem 45. The system $\mathsf{KAT}{+}H$ is complete for the interpretation $\mathcal{G}_h$.

Proof. Consider the tuple (P, Id, R, Z) of Example 36, modified so that Z contains additionally Z_h. To verify that is satisfies Assumption 35 we need to check seamlessness: for an undefined $\gamma p \delta \in Z_h$ we have that $\gamma\, \gamma p \delta \to_R \gamma p \delta$ and $\gamma p \delta\, \delta \to_R \gamma p \delta$. In fact, the tuple is well-behaved as seen in Example 42 (the set Z is still finite hence regular). Arguing as in the proof of Theorem 44, Theorem 43 gives us the desired completeness result. □

2.7.3 Redundant actions

We consider now equations of the form $c; p \equiv c$, where c is a test and p is an atomic action. We claim that $c; p \equiv c$ is equivalent to the conjunction of $\gamma; p \equiv \gamma$ for $\gamma \leq c$. Suppose that $c; p \equiv c$ and notice for $\gamma \leq c$ that $\gamma; p \equiv \gamma; c; p \equiv \gamma; c \equiv \gamma$. For the converse, we assume that $\gamma; p \equiv \gamma$ for every $\gamma \leq c$ and we show:

$$c; p \equiv (\textstyle\sum_{\gamma \leq c} \gamma); p \equiv \sum_{\gamma \leq c} \gamma; p \equiv \sum_{\gamma \leq c} \gamma \equiv c.$$

So, without loss of generality, we can restrict our attention to equations of the form $\gamma; p \equiv \gamma$, where γ is an atom, and p is an atomic action.

Let X be a finite set of strings of the form γp, where γ is an atom and p is an atomic action symbol, and H_r be the set of equations $\gamma; p \equiv \gamma$ for every γp in X. For an atomic action symbol p, define the set of atoms $A(p) = \{\gamma \mid \gamma p \in X\}$. Intuitively, $A(p)$ is the set of atoms after which it is redundant to execute the action p. Let $\mathcal{G}_r$ be the interpretation that differs from $\mathcal{G}_{\mathsf{KAT}}$ only for the base case of atomic action symbols:

$$\mathcal{G}_r(p) = A(p) \cup \{\gamma p \delta \mid \gamma \notin A(p) \text{ and } \delta \text{ is atom}\}.$$

We write $\mathsf{KAT}{+}H_r$ for the extension of KAT with extra axioms H_r.

Theorem 46. The system $\mathsf{KAT}{+}H_r$ is complete for the interpretation $\mathcal{G}_r$.

Proof. Fix an equation $\gamma; p \equiv \gamma$ in H_r. We claim that it is equivalent to the conjunction of the equations $\gamma; p; \gamma \equiv \gamma$ and $\gamma; p; \delta \equiv 0$ for $\delta \neq \gamma$. For the left-to-right direction, we assume $\gamma; p \equiv \gamma$ and show $\gamma; p; \gamma \equiv \gamma; \gamma \equiv \gamma$ and $\gamma; p; \delta \equiv \gamma; \delta \equiv 0$. For the right-to-left direction:

$$\gamma; p \equiv \gamma; p; (\gamma + \textstyle\sum_{\delta \neq \gamma} \delta) \equiv \gamma; p; \gamma + \sum_{\delta \neq \gamma} \gamma; p; \delta \equiv \gamma.$$

We define the tuple (P, Id, R, Z) so that R contains the base rules as well as

$\gamma p\gamma \to \gamma$ for every $\gamma p \in X$, and Z contains the standard undefined strings as well as $\gamma p\delta$ for every $\gamma p \in X$ and $\delta \neq \gamma$. First, we claim that the tuple satisfies Assumption 35. We examine critical pairs:

$$\begin{tikzcd}[row sep=small] \gamma\,\gamma p\gamma \arrow[r] \arrow[d] & \gamma\,\gamma \arrow[d] \\ \gamma p\gamma \arrow[r] & \gamma \end{tikzcd} \qquad \begin{tikzcd}[row sep=small] \gamma p\gamma\,\gamma \arrow[r] \arrow[d] & \gamma p\gamma \arrow[d] \\ \gamma\,\gamma \arrow[r] & \gamma \end{tikzcd}$$

It follows that R is confluent and terminating. For the seamlessness condition we simply verify that $\gamma\,\gamma p\delta \to_R \gamma p\delta$ and $\gamma p\delta\,\delta \to_R \gamma p\delta$. It remains to show that (P, Id, R, Z) is well-behaved. Fix an atom α and define $P_X(\alpha) = \{p \mid \alpha p \in X\}$. That is, $P_X(\alpha)$ is the set of all atomic actions that are redundant when α holds. The R-ancestors of α are

$$\mathsf{Ance}_R(\alpha) = \mathcal{R}(e_\alpha), \text{ where } e_\alpha = (\alpha + \textstyle\sum P_X(\alpha))^+.$$

We have already seen that $\mathsf{KA}_R \vdash \alpha \equiv \alpha^+$ (Example 21). So, reasoning in KA_R:

$$e_\alpha \equiv (\alpha + \textstyle\sum P_X(\alpha))^+ \equiv (\alpha + \alpha)^+ \equiv \alpha^+ \equiv \alpha.$$

For a letter $\alpha p\beta$ in Σ, its R-ancestors are $\mathsf{Ance}_R(\alpha p\beta) = e_{\alpha p\beta}$, where

$$e_{\alpha p\beta} = (\alpha + \textstyle\sum P_X(\alpha))^*; \alpha p\beta; (\beta + \textstyle\sum P_X(\beta))^*.$$

We also have that $\mathsf{KA}_R \vdash e_{\alpha p\beta} \equiv \alpha^*; \alpha p\beta; \beta^* \equiv \alpha p\beta$. We have thus verified well-behavedness and Theorem 43 says that $\mathsf{KA}_{\mathit{Id}}$ is complete for $\mathcal{G}_\perp$. But $\mathcal{G}_\perp$ is essentially the same as $\mathcal{G}_r$, and $\mathsf{KA}_{\mathit{Id}}$ is essentially the same as $\mathsf{KAT}+H_r$. $\square$

2.7.4 Mutable tests

Let Id be a finite set of atoms, and suppose that the atomic actions are $P = \{p_\alpha \mid \alpha \in \mathit{Id}\}$. The intuition is that p_α makes the atom α true. Consider axioms:

$$p_\alpha \equiv p_\alpha; \alpha \qquad \alpha; p_\alpha \equiv \alpha \qquad p_\alpha; p_\beta \equiv p_\beta$$

The first axiom says that α is true after executing p_α. The second axiom asserts that p_α is redundant when α is true. Finally, the third axiom says that p_β overrides the effect of p_α. These axioms together with KAT constitute a reduced presentation of the system $B!$, which is studied in [44]. Let us call this reduced system MutKAT. We define an interpretation $\mathcal{G}_m$, which sends an expression to a set of strings of the form $\alpha p_\beta \beta$.

$$\mathcal{G}_m(\alpha) = \{\alpha p_\alpha \alpha\} \qquad \mathcal{G}_m(p_\beta) = \{\alpha p_\beta \beta \mid \alpha \in \mathit{Id}\} \qquad \alpha p_\beta \beta \diamond \beta p_\gamma \gamma = \alpha p_\gamma \gamma$$

The coalesced product $\diamond$ is undefined for operands $\alpha p_\beta \beta$ and $\gamma p_\delta \delta$ with $\beta \neq \gamma$. A string $\alpha p_\beta \beta$ is essentially a pair (α, β) of atoms.

Theorem 47. The system MutKAT is complete for the interpretation $\mathcal{G}_m$.

Proof. First, we observe that MutKAT can be given equivalently by extending KAT with

$$\alpha; p_\alpha; a \equiv \alpha \qquad \alpha; p_\beta; \gamma \equiv 0, \text{ for } \beta \neq \gamma \qquad (\alpha; p_\beta; \beta); (\beta; p_\gamma; \gamma) \equiv \alpha; p_\gamma; \gamma$$

We define (P, Id, R, Z) with additional rules $\alpha p_\alpha \alpha \to \alpha$ and $\alpha p_\beta \beta\, \beta p_\gamma \gamma \to \alpha p_\gamma \gamma$, and Z with additional undefined strings $\alpha p_\beta \gamma$ where $\beta \neq \gamma$. We examine critical pairs:

$$\begin{tikzcd}
\alpha\, \alpha p_\alpha \alpha \arrow[r] \arrow[d] & \alpha\, \alpha \arrow[d] \\
\alpha p_\alpha \alpha \arrow[r] & \alpha
\end{tikzcd}
\qquad
\begin{tikzcd}
\alpha\, \alpha p_\beta \beta\, \beta p_\gamma \gamma \arrow[r] \arrow[d] & \alpha\, \alpha p_\gamma \gamma \arrow[d] \\
\alpha p_\beta \beta\, \beta p_\gamma \gamma \arrow[r] & \alpha p_\gamma \gamma
\end{tikzcd}
\qquad
\begin{tikzcd}
\alpha p_\alpha \alpha\, \alpha p_\beta \beta \arrow[d] \arrow[dr] & \\
\alpha\, \alpha p_\beta \beta \arrow[r] & \alpha p_\beta \beta
\end{tikzcd}$$

$$\begin{tikzcd}
\alpha p_\alpha \alpha\, \alpha \arrow[r] \arrow[d] & \alpha\, \alpha \arrow[d] \\
\alpha p_\alpha \alpha \arrow[r] & \alpha
\end{tikzcd}
\qquad
\begin{tikzcd}
\alpha p_\beta \beta\, \beta p_\gamma \gamma\, \gamma \arrow[r] \arrow[d] & \alpha p_\gamma \gamma\, \gamma \arrow[d] \\
\alpha p_\beta \beta\, \beta p_\gamma \gamma \arrow[r] & \alpha p_\gamma \gamma
\end{tikzcd}
\qquad
\begin{tikzcd}
\alpha p_\beta \beta\, \beta p_\beta \beta \arrow[d] \arrow[dr] & \\
\alpha p_\beta \beta\, \beta \arrow[r] & \alpha p_\beta \beta
\end{tikzcd}$$

$$\begin{tikzcd}
\alpha p_\beta \beta\, \beta p_\gamma \gamma\, \gamma p_\delta \delta \arrow[r] \arrow[d] & \alpha p_\beta \beta\, \beta p_\delta \delta \arrow[d] \\
\alpha p_\gamma \gamma\, \gamma p_\delta \delta \arrow[r] & \alpha p_\delta \delta
\end{tikzcd}$$

So, R is confluent and terminating. For seamlessness, we check the nontrivial cases:

$$\begin{array}{ll}
\text{for } \alpha \neq \beta : \underline{\alpha p_\alpha \alpha\, \beta} \to_R \underline{\alpha\, \beta} & \text{for } \alpha \neq \beta : \underline{\alpha\, \beta p_\gamma \gamma}\, \gamma p_\delta \delta \to_R \underline{\alpha\, \beta p_\delta \delta} \\
\text{for } \alpha \neq \beta : \underline{\alpha\, \beta p_\beta \beta} \to_R \underline{\alpha\, \beta} & \text{for } \gamma \neq \delta : \alpha p_\beta \beta\, \underline{\beta p_\gamma \gamma\, \delta} \to_R \underline{\alpha p_\gamma \gamma\, \delta} \\
\text{for } \alpha \neq \beta : \underline{\alpha p_\alpha \alpha\, \beta p_\gamma \gamma} \to_R \underline{\alpha\, \beta p_\gamma \gamma} & \text{for } \beta \neq \gamma : \underline{\alpha p_\beta \beta\, \gamma p_\delta \delta}\, \delta p_\zeta \zeta \to_R \underline{\alpha p_\beta \beta\, \gamma p_\zeta \zeta} \\
\text{for } \beta \neq \gamma : \underline{\alpha p_\beta \beta\, \gamma p_\gamma \gamma} \to_R \underline{\alpha p_\beta \beta\, \gamma} & \text{for } \gamma \neq \delta : \alpha p_\beta \beta\, \underline{\beta p_\gamma \gamma\, \delta p_\zeta \zeta} \to_R \underline{\alpha p_\gamma \gamma\, \delta p_\zeta \zeta}
\end{array}$$

We have underlined above the undefined substrings. First, we consider the R-ancestors of the letter $\alpha p_\beta \beta$, where $\alpha \neq \beta$. The claim is that:

$$\mathsf{Ance}_R(\alpha p_\beta \beta) = \mathcal{R}(e_{\alpha p_\beta \beta}), \text{ where}$$
$$e_{\alpha p_\beta \beta} = \mathsf{row}(\alpha); (M+I)^*; M; (M+I)^*; \mathsf{col}(\beta).$$

The expression $\mathsf{row}(\alpha)$ above is a matrix of type $1 \times \mathit{Id}$ (row vector) with 1 at the α-indexed location and 0 elsewhere. Similarly, the expression $\mathsf{col}(\beta)$ is a matrix of type $\mathit{Id} \times 1$ (column vector) with 1 at the β-indexed location and 0 elsewhere. The square matrices M and I are both of type $\mathit{Id} \times \mathit{Id}$. We define M and I as:

$$M(\gamma, \delta) = \gamma p_\delta \delta \qquad I(\gamma, \gamma) = \gamma \qquad I(\gamma, \delta) = 0, \text{ for } \gamma \neq \delta$$

Informally, the matrix M describes transitions from one atom to another, and I describes the ε-transitions. The intuition for the definition of the expression $e_{\alpha p_\beta \beta}$ is that it represents all paths from α to β with at least one "strict step" of the form $\gamma p_\delta \delta$. We define the $1 \times \mathit{Id}$ matrix N by $N(\gamma) = \alpha p_\gamma \gamma$ and we reason as follows in KA_R:

- Claim: $M + I \equiv M$. Notice that $(M+I)(\gamma,\gamma) \equiv \gamma p_\gamma \gamma + \gamma \equiv M(\gamma, \gamma)$ and for $\gamma \neq \delta$ we have that $(M+I)(\gamma, \delta) \equiv \gamma p_\delta \delta + 0 \equiv M(\gamma, \gamma)$.
- Claim: $\mathsf{row}(\alpha); M \equiv N$. It holds that $(\mathsf{row}(\alpha); M)(\gamma) \equiv M(\alpha, \gamma) \equiv \alpha p_\gamma \gamma \equiv N(\gamma)$.
- Claim: $\mathsf{row}(\alpha); M^* \equiv \mathsf{row}(\alpha) + N$. The inequality $\mathsf{row}(\alpha) + N \leq \mathsf{row}(\alpha); M^*$

follows easily from the previous claim and KA. It also holds that

$$\begin{aligned}\mathsf{row}(\alpha); M^* \leq \mathsf{row}(\alpha) + N &\Longleftarrow (\mathsf{row}(\alpha) + N); M \leq \mathsf{row}(\alpha) + N \\ &\Longleftarrow \mathsf{row}(\alpha); M + N; M \equiv \mathsf{row}(\alpha) + N \\ &\Longleftarrow N + N \leq \mathsf{row}(\alpha) + N,\end{aligned}$$

which is is provable.

– Claim: $N; M \equiv N$. This holds because $(N; M)(\gamma) \equiv \sum_\delta N(\delta); M(\delta, \gamma) \equiv \sum_\delta \alpha p_\delta \delta; \delta p_\gamma \gamma$, which is provably equal to $\alpha p_\gamma \gamma \equiv N(\gamma)$.

Using the claims we have just proved, we continue to reason in KA_R:

$$\begin{aligned}\mathsf{row}(\alpha); (M + I)^*; M; (M + I)^*; \mathsf{col}(\beta) &= \mathsf{row}(\alpha); M^*; M; M^*; \mathsf{col}(\beta) \\ &\equiv \mathsf{row}(\alpha); M^*; M; \mathsf{col}(\beta) \\ &\equiv (\mathsf{row}(\alpha) + N); M; \mathsf{col}(\beta) \\ &\equiv \mathsf{row}(\alpha); M; \mathsf{col}(\beta) + N; M; \mathsf{col}(\beta) \\ &\equiv N; \mathsf{col}(\beta) + N; \mathsf{col}(\beta),\end{aligned}$$

which is provably equal to $N(\beta) \equiv \alpha p_\beta \beta$. Now, the R-ancestors of an atom α are

$$\mathsf{Ance}_R(\alpha) = \mathcal{R}(e_\alpha), \text{ where } e_\alpha = \alpha^+ + e_{\alpha p_\alpha \alpha}$$

It also holds that $\mathsf{KA}_R \vdash e_\alpha \equiv \alpha^+ + e_{\alpha p_\alpha \alpha} \equiv \alpha + \alpha p_\alpha \alpha \equiv \alpha$. Finally, the R-ancestors of a letter $\alpha p_\beta \gamma$ with $\beta \neq \gamma$ are given as follows:

$$\mathsf{Ance}_R(\alpha p_\beta \gamma) = \mathcal{R}(e_{\alpha p_\beta \gamma}), \text{ where } e_{\alpha p_\beta \gamma} = (1 + e_\alpha); \alpha p_\beta \gamma; (1 + e_\gamma).$$

So, (P, Id, R, Z) is well-behaved and Theorem 43 says that $\mathsf{KA}_{\mathit{Id}}$ is complete for $\mathcal{G}_\perp$. Notice that $\mathcal{G}_\perp$ is essentially $\mathcal{G}_m$ and $\mathsf{KA}_{\mathit{Id}}$ is equivalent to MutKAT. □

2.7.5 NetKAT

The case of NetKAT [6] is an extension of MutKAT, which was studied previously in §2.7.4. We have a set Id of atoms, and the atomic actions are now

$$P = \{\mathsf{dup}\} \cup \{p_\alpha \mid \alpha \in Id\}.$$

So, the language of NetKAT extends the language of MutKAT with the additional atomic action dup that satisfies the axiom $\alpha; \mathsf{dup} = \mathsf{dup}; \alpha$. This axiom asserts that the action dup preserves the atom that currently holds true. Let $\mathcal{G}_n$ be the language interpretation for NetKAT that is defined in [6].

Theorem 48. The system NetKAT is complete for the interpretation $\mathcal{G}_n$.

Proof. NetKAT can be presented equivalently by extending KAT as follows:

$$\alpha; p_\alpha; a \equiv \alpha \qquad \alpha; p_\beta; \gamma \equiv 0, \text{ for } \beta \neq \gamma$$
$$(\alpha; p_\beta; \beta); (\beta; p_\gamma; \gamma) \equiv \alpha; p_\gamma; \gamma \qquad \alpha; \mathsf{dup}; \beta \equiv 0, \text{ for } \alpha \neq \beta$$

Define the rewrite system (P, Id, R, Z) as in the case of MutKAT (see proof of Theorem 47) with the only difference being that Z here contains additionally the strings $\alpha\mathsf{dup}\beta$ for atoms $\alpha \neq \beta$. In order the establish well-behavedness, the only extra obligations concern the R-ancestors of $\alpha\mathsf{dup}\beta$. Indeed, we have

$$\mathsf{Ance}_R(\alpha\mathsf{dup}\beta) = \mathcal{R}(e_{\alpha\mathsf{dup}\beta}), \text{ where } e_{\alpha\mathsf{dup}\beta} = (\alpha + \alpha p_\alpha \alpha)^*; \alpha\mathsf{dup}\beta; (\beta + \beta p_\beta \beta)^*.$$

Similarly to the proof of Theorem 47, it holds that $\mathsf{KA}_R \vdash e_{\alpha\mathsf{dup}\beta} = \alpha\mathsf{dup}\beta$. We can now invoke Theorem 43, which completes the proof. □

2.7.6 Test commutes with action

We consider the case of KAT with extra equations of the form $p; b \equiv b; p$, where b is a test and p is an atomic action. We claim that the equation $p; b \equiv b; p$ is equivalent to the conjunction of the following equations:

$$\beta; p; \gamma \equiv 0 \text{ (for } \beta \leq b\text{, and } \gamma \leq \neg b)$$
$$\gamma; p; \beta \equiv 0 \text{ (for } \gamma \leq \neg b\text{, and } \beta \leq b)$$

For one direction of the claim, we observe that:

$$p; b \equiv \textstyle\sum_{\gamma \in At,\, \beta \leq b} \gamma; p; \beta \equiv \left(\textstyle\sum_{\gamma \leq b,\, \beta \leq b} \gamma; p; \beta\right) + \textstyle\sum_{\gamma \leq \neg b,\, \beta \leq b} \gamma; p; \beta \equiv \textstyle\sum_{\gamma, \beta \leq b} \gamma; p; \beta$$
$$b; p \equiv \textstyle\sum_{\beta \leq b,\, \gamma \in At} \beta; p; \gamma \equiv \left(\textstyle\sum_{\beta \leq b,\, \gamma \leq b} \beta; p; \gamma\right) + \textstyle\sum_{\beta \leq b,\, \gamma \leq \neg b} \beta; p; \gamma \equiv \textstyle\sum_{\beta, \gamma \leq b} \beta; p; \gamma$$

It follows that $p; b \equiv b; p$. For the other direction of the claim, we have:

$$\text{for } \beta \leq b \text{ and } \gamma \leq \neg b : \beta; p; \gamma \leq b; p; \gamma \equiv p; b; \gamma \equiv 0 \implies \beta; p; \gamma \equiv 0$$
$$\text{for } \gamma \leq \neg b \text{ and } \beta \leq b : \gamma; p; \beta \leq \gamma; p; b \equiv \gamma; b; p \equiv 0 \implies \gamma; p; \beta \equiv 0$$

So, without loss of generality we can consider equations of the form $\beta; p; \gamma \equiv 0$, for atoms β, γ and atomic action p. This is exactly like the case of Hoare hypotheses, and so we obtain a completeness theorem as in §2.7.2.

2.8 Conclusion

We have identified sufficient conditions for the construction of free language models for systems of Kleene algebra with additional equations. The construction provides a uniform approach to deductive completeness and coalgebraic decision procedures (although we do not pursue this connection here). The criteria are given in terms of *inverse context-free rewrite systems* [23]. They imply the existence of free language models in a wide range of previously studied instances, including KAT [72] and NetKAT [6], as well as some new ones. We have also given a negative result that establishes a limit to the applicability of the technique.

For the future, we would like to investigate the possibility of developing a uniform approach to coalgebraic bisimulation-based decision procedures [6, 39, 44, 114, 22]. Such decision procedures typically involve some variant of Brzozowski derivatives and are highly dependent on the existence of language models.

CHAPTER 3

KAT WITH EXTRA MUTABLE TESTS

3.1 Introduction

Kleene algebra with tests (KAT) is a propositional equational system that combines Kleene algebra (KA) with Boolean algebra. It has been shown to be an effective tool for many low-level program analysis and verification tasks involving communication protocols, safety analysis, source-to-source program transformation, concurrency control, and compiler optimization [7, 14, 27, 28, 29, 72, 81]. A notable recent success is its adoption as a basis for NetKAT, a foundation for software-defined networks (SDN) [6].

One advantage of KAT is that it allows a clean separation of the theory of the domain of computation from the program restructuring operations. The former typically involves first-order reasoning, whereas the latter is typically propositional. It is often advantageous to separate the two, because the theory of the domain of computation may be highly undecidable. With KAT, one typically isolates the needed properties of the domain as premises in a Horn formula

$$s_1 = t_1 \wedge \cdots \wedge s_n = t_n \to s = t,$$

where the conclusion $s = t$ expresses a more complicated equivalence between (say) an unoptimized or unannotated version of a program and its optimized or annotated version. The premises are verified once and for all using the properties of the domain, and the conclusion is then verified propositionally in KAT under those assumptions.

Certain premises that arise frequently in practice can be incorporated as part of the theory using a technique known as *elimination of hypotheses*, in which Horn formulas with premises of a certain form can be reduced to the equational theory without loss of efficiency [27, 82, 45]. However, there are a few useful ones that cannot. In particular, it is known that there are certain program transformations that cannot be effected in pure KAT, but require extra structure. Two paradigmatic examples are the Böhm–Jacopini theorem [21] (see also [10, 98, 103, 112, 125]) and the folklore result that all while programs can be transformed to a program with a single while loop [46, 96].

The Böhm–Jacopini theorem states that every deterministic flowchart can be written as a while program. The construction is normally done at the first-order level and introduces auxiliary variables to remember values across computations. It has been shown that the construction is not possible without some kind of auxiliary structure of this type [10, 60, 61, 85].

Akin to the Böhm–Jacopini theorem, and often erroneously conflated with it, is the folklore theorem that every while program can be written with a single while loop. Like the proof of the Böhm–Jacopini theorem, the proof of [96] (as reported in [46]) is normally done at the first-order level and uses auxiliary variables. It was a commonly held belief that this result had no purely propositional proof [46], but a partial refutation of this view was given in [72] using a construction that foreshadows the construction of this chapter.

One can carry out these constructions in an uninterpreted first-order version of KAT called *schematic KAT* (SKAT) [7, 78], but as SKAT is undecidable in general [76], one would prefer a less radical extension.

We investigate here the minimal amount of structure that suffices to perform these transformations and show how to incorporate it in KAT without sacrificing deductive completeness or decidability. Our main results are:

- We show how to extend KAT with a set of independent *mutable tests*. The construction is done axiomatically with generators and additional equational axioms. We formulate the construction as a general *commutative coproduct* construction that satisfies a certain universality property. The generators are abstract *setters* of the form $t!$ and $\bar{t}!$ and *testers* $t?$ and $\bar{t}?$ for a test symbol t. We can think of these intuitively as operations that set and test the value of a Boolean variable, although we do not introduce any explicit notion of storage or variable assignment.
- We prove a representation theorem (Theorem 58) for the commutative coproduct of an arbitrary KAT K and a KAT of binary relations on a finite set, namely that it is isomorphic to a certain matrix algebra over K.
- As a corollary to the representation theorem, we show that the extension is *conservative*; that is, an arbitrary KAT K can be augmented with mutable tests without affecting the theory of K. This is captured formally by a general property of the commutative coproduct, namely *injectivity*. It is not known whether the coproduct of KATs is injective in general, but we show that it is injective if at least one of the two cofactors is a finite KAT, which is the case in our application.
- We show that the free mutable test algebra on generators t_i, $1 \leq i \leq n$, is isomorphic to the KAT of all binary relations on a set of 2^n states.
- We show that the equational theory of an arbitrary KAT K augmented with mutable tests is axiomatically reducible to the theory of K. In particular, the free KAT, augmented with mutable tests, is completely axioma-

tized by the KAT axioms plus the axioms for mutable tests.

- We demonstrate that the folklore result about while programs can be carried out in KAT with mutable tests.

Balbiani et al. [13] present a related system DL-PA (which stands for Dynamic Logic of Propositional Assignments), a variant of propositional dynamic logic (PDL) with mutable tests only. Their system corresponds most closely to our free mutable test algebra. However, the semantics of DL-PA is restricted to relational models.

Outline This chapter is organized as follows. In §3.2 we introduce the theory of mutable tests and prove that the free mutable test algebra on n generators is isomorphic to the KAT of all binary relations on a set of size 2^n. In §3.3 we introduce the commutative coproduct construction and prove our representation theorem for the commutative coproduct of an arbitrary KAT K and a finite relational KAT. In §3.4 we present our main completeness results, and we apply the theory to give an axiomatic treatment of a significant application involving program transformations, namely the folk theorem on while programs mentioned above. In §3.5 we present conclusions and open problems.

3.2 Mutable Tests

Let $T_n = \{t_1, t_2, \ldots, t_n\}$ be a set of primitive symbols, which should be thought of as representing distinct Boolean variables. We write $\Sigma_n = \{t!, \bar{t}! \mid t \in T_n\}$ for the set of action symbols that informally set and reset the Boolean variables. So, we think of $t!$ as saying "make t true" and of $\bar{t}!$ as saying "make t false". We also have corresponding tests $B_n = \{t?, \bar{t}? \mid t \in T_n\}$, where $t?$ performs the test "is t true?" and similarly $\bar{t}?$ performs the test "is t false?". In Figure 3.1 we introduce a finite collection Mut_n of equational axioms which capture the essential properties of mutable tests over T_n. Let us give some intuitive explanation for some of the Mut_n axioms:

- The equation $t!; t? = t!$ says that the action $t!$ makes a subsequent test $t?$ true.
- The equation $t?; t! = t?$ says that if $t?$ is already true, then the action $t!$ is redundant.
- The equation $t!; \bar{t}! = \bar{t}!$ says that setting a variable with $\bar{t}!$ overrides a previous assignment $t!$ to the same variable.
- The equations $s!; t! = t!; s!$ and $s!; t? = t?; s!$ say that actions and tests on different values are independent.

The theory $\mathsf{MutKAT}_n^=$ refers to the equational consequences of the above axioms along with the axioms of KAT on terms over Σ_n and B_n. The equations

$$t!; t! = t!; t?; t! = t!; t? = t! \qquad\qquad t!; \bar{t}? = t!; t?; \bar{t}? = t!; 0 = 0$$

follow from the axioms. So, $t!; t! = t!$ and $t!; \bar{t}? = 0$ are in $\mathsf{MutKAT}_n^=$.

Example 49 (Negation)**.** The equations $\neg t? = \bar{t}?$ and $\neg\bar{t}? = t?$ are in $\mathsf{MutKAT}_n^=$.

Proof. Using both KAT axioms and equations from Mut_n we get the sequences of equations:

$$\neg t? = \neg t?; 1 = \neg t?; (t? + \bar{t}?) = \neg t?; t? + \neg t?; \bar{t}? = 0 + \neg t?; \bar{t}? = \neg t?; \bar{t}?$$

$$\bar{t}? = \bar{t}?; 1 = \bar{t}?; (t? + \neg t?) = \bar{t}?; t? + \bar{t}?; \neg t? = 0 + \bar{t}?; \neg t? = \bar{t}?; \neg t?$$

Positive & negative literals	$t?;\bar{t}? = 0$	$t? + \bar{t}? = 1$
Redundant test	$t!;t? = t!$	$\bar{t}!;\bar{t}? = \bar{t}!$
Redundant action	$t?;t! = t?$	$\bar{t}?;\bar{t}! = \bar{t}?$
Variable overwriting	$t!;\bar{t}! = \bar{t}!$	$\bar{t}!;t! = t!$
Write-write commutation $(s \neq t)$	$s!;t! = t!;s!$	$s!;\bar{t}! = \bar{t}!;s!$
	$\bar{s}!;t! = t!;\bar{s}!$	$\bar{s}!;\bar{t}! = \bar{t}!;\bar{s}!$
Write-read commutation $(s \neq t)$	$s!;t? = t?;s!$	$s!;\bar{t}? = \bar{t}?;s!$
	$\bar{s}!;t? = t?;\bar{s}!$	$\bar{s}!;\bar{t}? = \bar{t}?;\bar{s}!$

Figure 3.1: The essential equational properties of mutable tests.

Since tests commute, we get that $\neg t? = \bar{t}?$. It follows that $\neg\neg t? = \neg\bar{t}?$ and hence $t? = \neg\bar{t}?$. □

A T_n*-assignment* is a map $\rho : T_n \to 2$, where $2 = \{0, 1\}$. So, ρ gives us for every variable t whether it is true ($\rho(t) = 1$) or false ($\rho(t) = 0$). Now, for such a function $\rho : T_n \to 2$ we write $\mathsf{test}(\rho)$ for the test that checks whether every variable has the value prescribed by ρ. Similarly, we write $\mathsf{set}(\rho)$ for the action that sets each variable so that its value is equal to the one given by ρ. More formally, we define

$$\mathsf{test}(\rho) \triangleq \ell_1?;\ell_2?;\cdots;\ell_n? \qquad\qquad \mathsf{set}(\rho) \triangleq \ell_1!;\ell_2!;\cdots;\ell_n!$$

where $\ell_i = t$ if $\rho(t_i) = 1$ and $\ell_i = \bar{t}$ if $\rho(t_i) = 0$. A term of the form $\mathsf{test}(\rho)$ is called a *complete test*, and similarly a term of the form $\mathsf{set}(\rho)$ is called a *complete assignment*.

Lemma 50 (Reduced Axioms)**.** Let ρ and σ be T_n-assignments. The following are consequences of Mut_n and KAT:

$$\sum_\rho \mathsf{test}(\rho) = 1 \qquad\qquad \mathsf{set}(\rho);\mathsf{test}(\rho) = \mathsf{set}(\rho)$$

$$\mathsf{test}(\rho);\mathsf{test}(\rho) = \mathsf{test}(\rho) \qquad\qquad \mathsf{test}(\rho);\mathsf{set}(\rho) = \mathsf{test}(\rho)$$

$$\mathsf{test}(\rho);\mathsf{test}(\sigma) = 0 \text{ when } \rho \neq \sigma \qquad\qquad \mathsf{set}(\rho);\mathsf{set}(\sigma) = \mathsf{set}(\sigma)$$

Proof. All the above properties follow easily from the axioms for mutable tests given in Figure 3.1. For example, let us derive the first equation:

$$\begin{aligned}
1 &= 1; 1; \cdots ; 1 && \text{[KA axioms]} \\
&= (t_1? + \bar{t}_1?); (t_2? + \bar{t}_2?); \cdots ; (t_n? + \bar{t}_n?) && \text{[axiom } t? + \bar{t}? = 1\text{]} \\
&= t_1?; t_2?; \cdots ; t_n? + \cdots + \bar{t}_1?; \bar{t}_2?; \cdots ; \bar{t}_n? && \text{[distributivity]} \\
&= \textstyle\sum_\rho \mathsf{test}(\rho) && \text{[same term]}
\end{aligned}$$

where ρ in the sum above ranges over all T_n-assignments. We omit the proofs for the rest. □

Now, we claim that every primitive test and action can be written equivalently using complete tests and complete assignments as base terms:

$$t? = \textstyle\sum_{\rho(t)=1} \mathsf{test}(\rho); \mathsf{set}(\rho) \qquad t! = \textstyle\sum_\rho \mathsf{test}(\rho); \mathsf{set}(\rho[t \mapsto 1]) \tag{3.1}$$

$$\bar{t}? = \textstyle\sum_{\rho(t)=0} \mathsf{test}(\rho); \mathsf{set}(\rho) \qquad \bar{t}! = \textstyle\sum_\rho \mathsf{test}(\rho); \mathsf{set}(\rho[t \mapsto 0]) \tag{3.2}$$

The above equations can be established easily using the properties of Lemma 50. For example,

$$\begin{aligned}
t! &= \left(\textstyle\sum_\rho \mathsf{test}(\rho)\right); t! = \textstyle\sum_\rho \mathsf{test}(\rho); t! = \textstyle\sum_\rho \mathsf{test}(\rho); \mathsf{set}(\rho); t! \\
&= \textstyle\sum_\rho \mathsf{test}(\rho); \mathsf{set}(\rho[t \mapsto 1]).
\end{aligned}$$

Deriving the rest of the equations is equally straightforward, and we thus omit the proofs.

We have seen so far that the complete tests and assignments can be written using the primitive tests and assignments, and vice versa. Lemma 50 says that the given set of *reduced axioms* (in terms of complete tests and assignments) follows from the original axioms. It is also the case that the original axioms follow from the reduced axioms. So, the original axiomatization Mut and the reduced axiomatization of Lemma 50 are equivalent given the axioms of KAT.

3.2.1 Free KAT Generated by Mutable Tests

Let K_n be the free KAT generated by the actions Σ_n and the tests B_n modulo the equational consequences of Mut_n and KAT. This algebra arises from a standard construction, which we outline here for the sake of completeness. First, we define the sets of test-terms $\mathsf{Trm}_{\mathsf{B}}(T_n)$ and (general) terms $\mathsf{Trm}(T_n)$, for which the containment $\mathsf{Trm}_{\mathsf{B}}(T_n) \subseteq \mathsf{Trm}(T_n)$ holds.

$$0,\,1 \in \mathsf{Trm}_{\mathsf{B}}(T_n) \qquad \frac{t \in T_n}{t?,\, \bar{t}? \in \mathsf{Trm}_{\mathsf{B}}(T_n)} \qquad \frac{p, q \in \mathsf{Trm}_{\mathsf{B}}(T_n)}{p+q,\ p;q,\ \neg p \in \mathsf{Trm}_{\mathsf{B}}(T_n)}$$

$$\frac{p \in \mathsf{Trm}_{\mathsf{B}}(T_n)}{p \in \mathsf{Trm}(T_n)} \qquad \frac{t \in T_n}{t!,\, \bar{t}! \in \mathsf{Trm}(T_n)} \qquad \frac{f, g \in \mathsf{Trm}(T_n)}{f+g,\ f;g,\ f^* \in \mathsf{Trm}(T_n)}$$

Recall that we defined $\mathsf{MutKAT}_n^=$ to be the smallest set of equations that contains the Mut_n axioms of Figure 3.1 and is closed under the axioms and rules of KAT and Horn-equational logic. It follows that the equations of $\mathsf{MutKAT}_n^=$ describe a KAT-congruence on $\mathsf{Trm}(T_n)$. We denote by $[f]$ the congruence class of a term f in $\mathsf{Trm}(T_n)$. Now, we define the carriers $B_n \subseteq K_n$ as follows:

$$K_n \triangleq \mathsf{Trm}(T_n)/\mathsf{MutKAT}_n^= = \{[f] \mid f \in \mathsf{Trm}(T_n)\}$$

$$B_n \triangleq \mathsf{Trm}_{\mathsf{B}}(T_n)/\mathsf{MutKAT}_n^= = \{[p] \mid p \in \mathsf{Trm}_{\mathsf{B}}(T_n)\}$$

Since $\mathsf{MutKAT}_n^=$ describes a KAT-congruence, we can define the KAT operations on the congruence classes:

$$0_n \triangleq [0] \qquad [f]+[g] \triangleq [f+g] \qquad [f]^* \triangleq [f^*]$$

$$1_n \triangleq [1] \qquad [f];[g] \triangleq [f;g] \qquad \neg[p] \triangleq [\neg p]$$

The algebra $(K_n, B_n, +, ;, ^*, 0_n, 1_n, \neg)$ is called the *free KAT over mutable tests* T_n.

Lemma 51 (Canonical Forms)**.** For every term f in $\mathsf{Trm}(T_n)$ there is a finite collection $(\rho_i, \sigma_i)_i$ of pairs of T_n-assignments such that the equation

$$f = \textstyle\sum_i \mathsf{test}(\rho_i); \mathsf{set}(\sigma_i)$$

belongs to the equational theory $\mathsf{MutKAT}_n^=$.

Proof. By Boolean reasoning we know that every test p can be rewritten equivalently so that negation only appears in front of the primitive tests $t?$ and $\bar{t}?$. Since $\neg t? = \bar{t}?$ and $\neg \bar{t}? = t?$ are provable (see Example 49), we can assume without loss of generality that negation does not appear at all in the terms, except in the form of an overbar on primitive symbols

Now, the proof proceeds by induction on the structure of terms. The base cases $t?$, $\bar{t}?$, $t!$ and $\bar{t}!$ have already been handled in Equations (3.1) and (3.2). Moreover, 0 is the empty sum and

$$1 = \textstyle\sum_\rho \mathsf{test}(\rho) = \sum_\rho \mathsf{test}(\rho); \mathsf{set}(\rho)$$

is provable by Lemma 50. The step case for choice $+$ is trivial. For the case of composition ; we first observe (using equations of Lemma 50) that

$$\begin{array}{ll}
\mathsf{test}(\rho); \mathsf{set}(\sigma); \mathsf{test}(\sigma); \mathsf{set}(\tau) = & \mathsf{test}(\rho); \mathsf{set}(\sigma); \mathsf{test}(\tau); \mathsf{set}(\upsilon) = \\
\mathsf{test}(\rho); \mathsf{set}(\sigma); \mathsf{set}(\tau) = & \mathsf{test}(\rho); \mathsf{set}(\sigma); \mathsf{test}(\sigma); \mathsf{test}(\tau); \mathsf{set}(\upsilon) = \\
\mathsf{test}(\rho); \mathsf{set}(\tau) & \mathsf{test}(\rho); \mathsf{set}(\sigma); 0; \mathsf{set}(\upsilon) = 0 \text{ (when } \sigma \neq \tau)
\end{array}$$

are provable, hence using distributivity we are done. Finally, for the case of iteration, we observe that every term of the form f^* can be written equivalently as a finite sum $1 + f + f^2 + \cdots + f^m$ for some m, since there are finitely many T_n-assignments. By the previous cases, we are done. □

Definition 52 (Relational KAT With Mutable Tests)**.** Let S be a nonempty set. We represent a binary relation on S as a function of type $S \to \mathcal{P}(S)$, where $\mathcal{P}(S)$ is the powerset of S. We use letters $\phi, \psi, \chi, \ldots$ to range over binary relations and $\rho, \sigma, \tau, \ldots$ to range over elements of S. We define for every $\sigma \in S$ the test $\sigma?$ and the action $\sigma!$ as follows:

$$\sigma?(\rho) \triangleq \emptyset \text{ if } \rho \neq \sigma \qquad \sigma?(\sigma) \triangleq \{\sigma\} \qquad \sigma!(\rho) \triangleq \{\sigma\}$$

The constants 0_S and 1_S are given by $0_S(\rho) = \emptyset$ and $1_S(\rho) = \{\rho\}$. The operations of composition ;, binary choice $+$, arbitrary choice $\sum$, and iteration * are defined in the usual way:

$$(\phi;\psi)(\rho) \triangleq \bigcup_{\sigma\in\phi(\rho)}\psi(\sigma) \qquad (\textstyle\sum_i \phi_i)(\rho) \triangleq \bigcup_i \phi_i(\rho)$$

$$\phi+\psi \triangleq \textstyle\sum\{\phi,\psi\} \qquad \phi^* \triangleq \sum_{n\geq 0}\phi^n$$

where ϕ^n is the n-fold composite of ϕ, that is, $\phi^0 = 1_S$ and $\phi^{n+1} = \phi^n;\phi$. For a relation $\phi \leq 1_S$, define $\neg\phi \leq 1_S$ to be the unique relation satisfying $\phi + \neg\phi = 1_S$ and $\phi;\neg\phi = 0_S$. The set $S \to \mathcal{P}(S)$ together with the above operations forms a KAT of binary relations.

Consider now the particular case where S is equal to the set $T_n \to 2$ of T_n-assignments. We define for every $t \in T_n$ the tests $(t?)_S$ and $(\bar{t}?)_S$ as well as the actions $(t!)_S$ and $(\bar{t}!)_S$ as follows:

$$(t!)_S(\rho) \triangleq \{\rho[t\mapsto 1]\} \qquad (\bar{t}!)_S(\rho) \triangleq \{\rho[t\mapsto 0]\}$$

$$(t?)_S(\rho) \triangleq \begin{cases}\{\rho\}, & \text{if } \rho(t)=1\\ \emptyset, & \text{if } \rho(t)=0\end{cases} \qquad (\bar{t}?)_S(\rho) \triangleq \begin{cases}\emptyset, & \text{if } \rho(t)=1\\ \{\rho\}, & \text{if } \rho(t)=0\end{cases}$$

For elements ρ and σ in S, notice that the relation $\rho?;\sigma! : S \to \mathcal{P}(S)$ is the smallest relation that contains the pair (ρ,σ). More formally, $(\rho?;\sigma!)(\rho) = \{\sigma\}$ and $(\rho?;\sigma!)(\rho') = \emptyset$ when $\rho' \neq \rho$. Moreover, every relation $\phi : S \to \mathcal{P}(S)$ can be written as a sum $\phi = \sum_{\sigma\in\phi(\rho)} \rho?;\sigma!$.

Call K_S the KAT described in the above definition. We remark that K_S is isomorphic to the algebra $\mathsf{Mat}(S,2)$ of $(S\times S)$-indexed matrices with Boolean values. The composition operation in $\mathsf{Mat}(S,2)$ is matrix multiplication, choice $+$ is componentwise addition, and * corresponds to reflexive transitive closure.

Theorem 53 (Representation)**.** The free KAT K_n over the mutable tests T_n is isomorphic to the KAT of all binary relations on a set of size 2^n.

Proof. We write S as abbreviation for the set $T_n \to 2$ of T_n-assignments, and we denote by K_S the KAT of all binary relations on S (see Definition 52). Define $H : \mathsf{Trm}(T_n) \to K_S$ by:

$$
\begin{array}{lll}
H(0) = 0_S & H(1) = 1_S & H(f;g) = H(f);H(g) \\
H(t?) = (t?)_S & H(\bar{t}?) = (\bar{t}?)_S & H(f+g) = H(f)+H(g) \\
H(t!) = (t!)_S & H(\bar{t}!) = (\bar{t}!)_S & H(f^*) = H(f)^*
\end{array}
$$

We claim now that for every equation $f = g$ in the theory $\mathsf{MutKAT}_n^=$ we have $H(f) = H(g)$. This is shown by induction on the construction of the theory $\mathsf{MutKAT}_n^=$, and it hinges on the fact that K_S is a KAT and the distinguished elements $(t?)_S$, $(\bar{t}?)_S$, $(t!)_S$ and $(\bar{t}!)_S$ of the algebra satisfy the equations Mut_n of Figure 3.1. For example, the equation $t!;t? = t!$ is satisfied because

$$((t!)_S;(t?)_S)(\rho) = (t?)_S(\rho[t \mapsto 1]) = \{\rho[t \mapsto 1]\} \quad \text{and} \quad (t!)_S(\rho) = \{\rho[t \mapsto 1]\}$$

for every T_n-assignment ρ. Since H agrees on terms f, g for which $f = g$ is in $\mathsf{MutKAT}_n^=$, we can define $H_1 : \mathsf{Trm}(T_n)/\mathsf{MutKAT}_n^= \to K_S$ on the equivalence classes by $H_1([f]) = H(f)$. In fact, H_1 is a KAT homomorphism from K_n to K_S that also preserves the distinguished constants for the symbols $t?$, $\bar{t}?$, $t!$, and $\bar{t}!$. For example, commutation with $+$ is shown by

$$H_1([f]+[g]) = H_1([f+g]) = H(f+g) = H(f)+H(g) = H_1([f]) + H_1([g]),$$

and the rest of the operations are handled similarly.

Now, we wish to show that the homomorphism $H_1 : K_n \to K_S$ is surjective. For this, notice that every relation $\phi : S \to \mathcal{P}(S)$ can be written as a finite sum

$$
\begin{aligned}
\phi &= \textstyle\sum_{\sigma \in \phi(\rho)} \rho?;\sigma! \\
&= \textstyle\sum_{\sigma \in \phi(\rho)} H(\mathsf{test}(\rho)); H(\mathsf{set}(\sigma)) \\
&= H\left(\textstyle\sum_{\sigma \in \phi(\rho)} \mathsf{test}(\rho); \mathsf{set}(\sigma)\right),
\end{aligned}
$$

because for the relations $\rho?$ and $\sigma!$ (Definition 52) we have $\rho? = H(\mathsf{test}(\rho))$ and $\sigma! = H(\mathsf{set}(\sigma))$.

Finally, the map $H_1 : K_n \to K_S$ is injective by virtue of Lemma 51, which gives us provably equivalent canonical forms. Let f and g be arbitrary terms. We also assume that $H_1([f])$ and $H_1([g])$ are equal. Lemma 51 says that there exist finite collections $(\rho_i, \sigma_i)_i$ and $(\tau_j, \upsilon_j)_j$ of pairs of T_n-assignments such that the equations

$$f = \textstyle\sum_i \mathsf{test}(\rho_i); \mathsf{set}(\sigma_i) \qquad \text{and} \qquad g = \textstyle\sum_j \mathsf{test}(\tau_j); \mathsf{set}(\upsilon_j)$$

belong to the theory $\mathsf{MutKAT}_n^=$. From the hypothesis $H_1([f]) = H_1([g])$ we obtain

$$\begin{aligned} H(f) = H(g) &\implies \textstyle\sum_i H(\mathsf{test}(\rho_i); \mathsf{set}(\sigma_i)) = \textstyle\sum_j H(\mathsf{test}(\tau_j); \mathsf{set}(\upsilon_j)) \\ &\implies \textstyle\sum_i \rho_i?; \sigma_i! = \textstyle\sum_j \tau_j?; \upsilon_j!, \end{aligned}$$

which implies that the sets $\{(\rho_i, \sigma_i) \mid i\}$ and $\{(\tau_j, \upsilon_j) \mid j\}$ are equal. So, with simple reasoning in KAT, we see that the terms $\sum_i \mathsf{test}(\rho_i); \mathsf{set}(\sigma_i)$ and $\sum_j \mathsf{test}(\tau_j); \mathsf{set}(\upsilon_j)$ are provably equivalent. It follows that the equation $f = g$ is in $\mathsf{MutKAT}_n^=$, and therefore $[f] = [g]$. $\square$

Corollary 54 (Completeness)**.** Let K_S be the KAT of all binary relation on the finite set S of T_n-assignments. The system $\mathsf{KAT}{+}\mathsf{Mut}_n$ is complete for the equational theory of K_S. In other words, if the terms f and g of $\mathsf{Trm}(T_n)$ denote the same relation in K_S (under the canonical interpretation), then the equation $f = g$ is provable.

Proof. The canonical interpretation $H : \mathsf{Trm}(T_n) \to K_S$ sends $t?$ to $(t?)_S$, and so on. For terms f and g, if $H(f) = H(g)$ then we have by the isomorphism $K_S \cong K_n$ (Theorem 53) that $[f] = [g]$. So, the equation $f = g$ belongs to $\mathsf{MutKAT}_n^=$, and is therefore provable in $\mathsf{KAT}{+}\mathsf{Mut}_n$. $\square$

3.3 The Commutative Coproduct of KATs

In this section we will present our central *commutative coproduct* construction. Let $(K, B, +, ;, {}^*, 0_K, 1_K, \neg)$ be an arbitrary KAT. For every element $x \in K$ we introduce a constant symbol c_x. We define the set of test-terms $\mathsf{Trm}(B)$ and (general) terms $\mathsf{Trm}(K)$ as follows:

$$0,\ 1 \in \mathsf{Trm}(B) \qquad \frac{b \in B}{c_b \in \mathsf{Trm}(B)} \qquad \frac{p, q \in \mathsf{Trm}(B)}{p + q,\ p;q,\ \neg p \in \mathsf{Trm}(B)}$$

$$\frac{x \in K}{c_x \in \mathsf{Trm}(K)} \qquad \frac{f, g \in \mathsf{Trm}(K)}{f + g,\ f;g,\ f^* \in \mathsf{Trm}(K)}$$

Of course, the containment $\mathsf{Trm}(B) \subseteq \mathsf{Trm}(K)$ holds, because we have that $B \subseteq K$. The function $H : \mathsf{Trm}(K) \to K$ is the unique homomorphism from $\mathsf{Trm}(K)$ to K given by $H : c_x \mapsto x$ for every element $x \in K$. The *diagram* of K, denoted Δ_K, is the set of equations between terms in $\mathsf{Trm}(K)$ that hold in K. That is,

$$\Delta_K \triangleq \{f = g \mid f, g \in \mathsf{Trm}(K) \text{ and } H(f) = H(g) \text{ in } K\}.$$

In other words, Δ_K is the kernel of the homomorphism H. It follows from general considerations of universal algebra that $\mathsf{Trm}(K)/\Delta_K \cong K$.

Definition 55 (Commutative Coproduct of KATs)**.** Let K and K' be KATs, which without loss of generality have disjoint carriers. Suppose that their test carriers are B and B' respectively. We write $\mathsf{Trm}(K, K')$ for the set of mixed terms over the union of the carriers of K and K', and similarly $\mathsf{Trm}(B, B')$ is the set of test-terms over $B \cup B'$.

$$0,\ 1 \in \mathsf{Trm}(B, B') \qquad \frac{b \in B \cup B'}{c_b \in \mathsf{Trm}(B, B')} \qquad \frac{p, q \in \mathsf{Trm}(B, B')}{p + q,\ p;q,\ \neg p \in \mathsf{Trm}(B, B')}$$

$$\frac{x \in K \cup K'}{c_x \in \mathsf{Trm}(K, K')} \qquad \frac{f, g \in \mathsf{Trm}(K, K')}{f + g,\ f;g,\ f^* \in \mathsf{Trm}(K, K')}$$

The above definitions imply that the following containments hold:

$$\mathsf{Trm}(B, B') \subseteq \mathsf{Trm}(K, K') \qquad \mathsf{Trm}(B), \mathsf{Trm}(B') \subseteq \mathsf{Trm}(B, B')$$
$$\mathsf{Trm}(K), \mathsf{Trm}(K') \subseteq \mathsf{Trm}(K, K')$$

The sets Δ_K and $\Delta_{K'}$ are the diagrams of K and K' respectively. We also define

$$D \triangleq \{c_x; c_y = c_y; c_x \mid x \in K \text{ and } y \in K'\}.$$

We call D the set of *commutativity equations*, which say informally that the elements of K and K' commute with respect to composition. Now, let $E_{K,K'}$ be the smallest set of equations between terms of $\mathsf{Trm}(K, K')$ that contains the diagrams Δ_K and $\Delta_{K'}$ and the commutativity equations D, and that is additionally closed under the axioms and rules of KAT and Horn-equational logic. Now, we define the *commutative coproduct* $K \boxplus K'$ of K and K' to be the quotient algebra

$$K \boxplus K' \triangleq \mathsf{Trm}(K, K')/E_{K,K'}.$$

Of course, $K \boxplus K'$ is a KAT, since the set $E_{K,K'}$ of equations contains all instances of equational KAT axioms and is closed under all instances of KAT implications.

We define the maps *left coprojection* $\iota_1 : K \to K \boxplus K'$ and *right coprojection* $\iota_2 : K' \to K \boxplus K'$ by $\iota_1(x) = [c_x]$ and $\iota_2(u) = [c_u]$, where $[c_x]$ is the congruence class of c_x in $K \boxplus K'$. It is easily verified that both ι_1 and ι_2 are KAT homomorphisms. For example, we have

$$\iota_1(x; y) = [c_{x;y}] = [c_x; c_y] = [c_x]; [c_y] = \iota_1(x); \iota_1(y),$$

because the equation $c_x; c_y = c_{x;y}$ is in the diagram $\Delta_K \subseteq E_{K,K'}$. Moreover,

$$\iota_1(x); \iota_2(y) = [c_x]; [c_y] = [c_x; c_y] = [c_y; c_x] = [c_y]; [c_x] = \iota_2(y); \iota_1(x)$$

for $x \in K$ and $y \in K'$, because the commutativity equation $c_x; c_y = c_y; c_x$ is in D.

Lemma 56 (Canonical Forms)**.** Let K be a KAT and F be a finite KAT. Every element of $K \boxplus F$ can be expressed as a finite sum

$$\textstyle\sum_{u \in F} \iota_1(x(u)); \iota_2(u)$$

for some function $x : F \to K$.

Proof. Every element of $K \boxplus F$ is of the form $[f]$ for some term $f \in \mathsf{Trm}(K, F)$. The proof is by induction on the structure of the term f. The base cases c_x for $x \in K$ and c_u for $u \in F$ are trivial. The case of $+$ is easy, and for ; we use commutativity (equations in D) and distributivity. The case of negation does not need to be handled separately, because $\neg$ can be pushed down to the leaves of the terms and $\neg[c_b] = [c_{\neg b}]$ in $K \boxplus F$ for a test b.

Finally, we handle the case of iteration $[f^*] = [f]^*$, which is the most interesting case. The induction hypothesis gives us that the element $[f] \in K \boxplus F$ is of the form

$$[f] = \textstyle\sum_{u \in F} \iota_1(x(u)); \iota_2(u)$$

for some $x : F \to K$. Consider the finite alphabet $\Sigma_F = \{a_u \mid u \in F\}$, and define the functions:

$$\begin{array}{lll} g : \Sigma_F \to K \boxplus F & h : \Sigma_F \to K & \iota_1 \circ h : \Sigma_F \to K \boxplus F \\ \quad a_u \mapsto \iota_1(x(u)); \iota_2(u) & \quad a_u \mapsto x(u) & \quad a_u \mapsto \iota_1(x(u)) \end{array}$$

Let $\mathsf{Reg}(\Sigma_F)$ be the algebra of regular sets over Σ_F, which is the free KA on generators Σ_F [69, 70]. Since both $K \boxplus F$ and K are Kleene algebras and ι_1 is a KA-homomorphism, the functions g, h, and $\iota_1 \circ h$ extend to KA-homomorphisms:

$$g : \mathsf{Reg}(\Sigma_F) \to K \boxplus F \qquad h : \mathsf{Reg}(\Sigma_F) \to K \qquad \iota_1 \circ h : \mathsf{Reg}(\Sigma_F) \to K \boxplus F$$

We know that a KA-homomorphism lifts to a KA-homomorphism between the matrix algebras:

$$\begin{aligned}
\bar{g} &: \mathsf{Mat}(F, \mathsf{Reg}(\Sigma_F)) \to \mathsf{Mat}(F, K \boxplus F) & \bar{g}(M)_{st} &\triangleq g(M_{st}) \\
\bar{h} &: \mathsf{Mat}(F, \mathsf{Reg}(\Sigma_F)) \to \mathsf{Mat}(F, K) & \bar{h}(M)_{st} &\triangleq h(M_{st}) \\
\bar{\iota}_1 &: \mathsf{Mat}(F, K) \to \mathsf{Mat}(F, K \boxplus F) & \bar{\iota}_1(M)_{st} &\triangleq \iota_1(M_{st})
\end{aligned}$$

where $\mathsf{Mat}(F, K)$ is the KA of $(F \times F)$-matrices over K. Define the matrix A by

$$A_{st} \triangleq \textstyle\sum\{a_u \mid u \in F \text{ and } s;u = t \text{ in } F\} \text{ for } s, t \text{ in } F.$$

We consider A to be an element of $\mathsf{Mat}(F, \mathsf{Reg}(\Sigma_F))$ (we have slightly abused notation above). We can also think of A as representing an automaton with states F and having a a_u-labeled transition $s \to t$ for every u with $s;u = t$. It follows that:

$$\begin{aligned}
\bar{g}(A)_{st} &= g(A_{st}) = g(\textstyle\sum_{s;u=t} a_u) = \sum_{s;u=t} g(a_u) = \sum_{s;u=t} \iota_1(x(u)); \iota_2(u) \\
\bar{h}(A)_{st} &= h(A_{st}) = h(\textstyle\sum_{s;u=t} a_u) = \sum_{s;u=t} h(a_u) = \sum_{s;u=t} x(u) \\
(\bar{\iota}_1 \circ \bar{h})(A)_{st} &= \bar{\iota}_1(\bar{h}(A))_{st} = \iota_1(\bar{h}(A)_{st}) = \iota_1(h(A_{st})) = \textstyle\sum_{s;u=t} \iota_1(x(u))
\end{aligned}$$

We also define the matrix M in $\mathsf{Mat}(F, K \boxplus F)$ as follows:

$$M_{ss} \triangleq \iota_2(u) \qquad\qquad M_{st} \triangleq 0_{K \boxplus F} \text{ for } s \neq t$$

We claim that $M \cdot \bar{g}(A) = (\bar{\iota}_1 \circ \bar{h})(A) \cdot M$. Indeed, for all s and t in F we have:

$$\begin{aligned}
(M \cdot \bar{g}(A))_{st} &= \textstyle\sum_{u \in F} M_{su}; \bar{g}(A)_{ut} && \text{[matrix multiplication]} \\
&= M_{ss}; \bar{g}(A)_{st} && \text{[definition of } M\text{]} \\
&= \iota_2(s); \textstyle\sum_{s;u=t} \iota_1(x(u)); \iota_2(u) && \text{[definition of } M \text{ and } \bar{g}(A)\text{]} \\
&= \textstyle\sum_{s;u=t} \iota_2(s); \iota_1(x(u)); \iota_2(u) && \text{[distributivity]} \\
&= \textstyle\sum_{s;u=t} \iota_1(x(u)); \iota_2(s); \iota_2(u) && [c_s; c_{x(u)} = c_{x(u)}; c_s \text{ in } D \subseteq E_{K,F}] \\
&= \textstyle\sum_{s;u=t} \iota_1(x(u)); \iota_2(t) && [c_s; c_u = c_t \text{ in } \Delta_F \subseteq E_{K,F}] \\
((\bar{\iota}_1 \circ \bar{h})(A) \cdot M)_{st} &= \textstyle\sum_{u \in F} (\bar{\iota}_1 \circ \bar{h})(A)_{su}; M_{ut} && \text{[matrix multiplication]} \\
&= (\bar{\iota}_1 \circ \bar{h})(A)_{st}; M_{tt} && \text{[definition of } M\text{]}
\end{aligned}$$

$$= \left(\textstyle\sum_{s;u=t} \iota_1(x(u))\right); \iota_2(t) \qquad \text{[def. of } (\bar{\iota}_1 \circ \bar{h})(A) \text{ and } M]$$

$$= \textstyle\sum_{s;u=t} \iota_1(x(u)); \iota_2(t) \qquad \text{[distributivity]}$$

Since $\mathsf{Mat}(F, K \boxplus F)$ is a KA, we have by the bisimulation rule (1.3) of Kleene algebra the following implications:

$$M \cdot \bar{g}(A) = (\bar{\iota}_1 \circ \bar{h})(A) \cdot M \implies \qquad \text{[bismulation rule]}$$

$$M \cdot \bar{g}(A)^* = (\bar{\iota}_1 \circ \bar{h})(A)^* \cdot M \implies \qquad \text{[homomorphisms]}$$

$$M \cdot \bar{g}(A^*) = (\bar{\iota}_1 \circ \bar{h})(A^*) \cdot M \implies \qquad \text{[matrices]}$$

$$(M \cdot \bar{g}(A^*))_{st} = ((\bar{\iota}_1 \circ \bar{h})(A^*) \cdot M)_{st} \text{ for all } s, t \in F \implies \qquad \text{[definition of } M]$$

$$M_{ss}; \bar{g}(A^*)_{st} = (\bar{\iota}_1 \circ \bar{h})(A^*)_{st}; M_{tt} \text{ for all } s, t \in F \implies \qquad \text{[definition of } M]$$

$$\iota_2(s); \bar{g}(A^*)_{st} = (\bar{\iota}_1 \circ \bar{h})(A^*)_{st}; \iota_2(t) \text{ for all } s, t \in F.$$

We instantiate the above equation for $s = 1_F$ and get

$$\bar{g}(A^*)_{1t} = (\bar{\iota}_1 \circ \bar{h})(A^*)_{1t}; \iota_2(t).$$

Now, we have that

$$[f]^* = (\textstyle\sum_{u \in F} \iota_1(x(u)); \iota_2(u))^* \qquad \text{[induction hypothesis]}$$

$$= (\textstyle\sum_{u \in F} g(a_u))^* \qquad \text{[definition of } g]$$

$$= g((\textstyle\sum_{u \in F} a_u)^*) \qquad [g \text{ homomorphism}]$$

$$= g(\textstyle\sum_{t \in F} (A^*)_{1t}) \qquad \text{[automaton } A]$$

$$= \textstyle\sum_{t \in F} g((A^*)_{1t}) \qquad [g \text{ homomorphism}]$$

$$= \textstyle\sum_{t \in F} \bar{g}(A^*)_{1t} \qquad \text{[definition of } \bar{g}]$$

$$= \textstyle\sum_{t \in F} (\bar{\iota}_1 \circ \bar{h})(A^*)_{1t}; \iota_2(t), \qquad \text{[see claim above]}$$

$$= \textstyle\sum_{t \in F} \iota_1(\bar{h}(A^*)_{1t}); \iota_2(t). \qquad [\iota_1 \text{ on matrices}]$$

Since the final sum is of the desired form

$$\textstyle\sum_{t \in F} \iota_1(\bar{h}(A^*)_{1t}); \iota_2(t) = \textstyle\sum_{t \in F} \iota_1(y(t)); \iota_2(t)$$

where $y : F \to K$ is given by $y(t) = \bar{h}(A^*)_{1t}$, we are done. $\square$

In the following lemma, we specialize the setting of Lemma 56 by considering the coproduct with a finite KAT of all binary relations over a finite set.

Lemma 57 (Canonical Forms)**.** Let K be a KAT and F be the KAT of all binary relations over a finite set S (see Definition 52). Every element of $K \boxplus F$ can be expressed as a finite sum

$$\textstyle\sum_{\rho,\sigma\in S} \iota_1(z(\rho,\sigma)); \iota_2(\rho?;\sigma!)$$

for some function $z : S \times S \to K$.

Proof. From Lemma 56 we know that every element of $K \boxplus F$ is of the form

$$\textstyle\sum_{\phi\in F} \iota_1(y(\phi)); \iota_2(\phi)$$

for some function $y : F \to K$. For every relation $\phi \in F$ we define the map $x_\phi : S \times S \to \{0_F, 1_F\}$ by $x_\phi(\rho,\sigma) = 1_F$ iff $\rho?;\sigma! \leq \phi$. It is then easy to see that ϕ can be written as a finite sum

$$\phi = \textstyle\sum_{\rho,\sigma} x_\phi(\rho,\sigma); \rho?; \sigma!.$$

Since the coprojections $\iota_1 : K \to K \boxplus F$ and $\iota_2 : F \to K \boxplus F$ are KAT homomorphisms, we have:

$$\begin{aligned}
&\textstyle\sum_{\phi\in F} \iota_1(y(\phi)); \iota_2(\phi) = && \text{[expand } \phi\text{]}\\
&\textstyle\sum_{\phi\in F} \iota_1(y(\phi)); \iota_2\left(\sum_{\rho,\sigma} x_\phi(\rho,\sigma); \rho?;\sigma!\right) = && [\iota_2 \text{ homomorphism}]\\
&\textstyle\sum_{\phi\in F} \iota_1(y(\phi)); \sum_{\rho,\sigma} \iota_2(x_\phi(\rho,\sigma)); \iota_2(\rho?;\sigma!) = && \text{[distributivity]}\\
&\textstyle\sum_{\phi\in F} \sum_{\rho,\sigma} \iota_1(y(\phi)); \iota_2(x_\phi(\rho,\sigma)); \iota_2(\rho?;\sigma!) = && \text{[rearrange sum]}\\
&\textstyle\sum_{\rho,\sigma} \sum_{\phi\in F} \iota_1(y(\phi)); \iota_2(x_\phi(\rho,\sigma)); \iota_2(\rho?;\sigma!) = && \text{[distributivity]}\\
&\textstyle\sum_{\rho,\sigma} \left(\sum_{\phi\in F} \iota_1(y(\phi)); \iota_2(x_\phi(\rho,\sigma))\right); \iota_2(\rho?;\sigma!) = && [\iota_1 \text{ homomorphism}]\\
&\textstyle\sum_{\rho,\sigma} \iota_1(z(\rho,\sigma)); \iota_2(\rho?;\sigma!), &&
\end{aligned}$$

where $z(\rho,\sigma) = \sum\{y(\phi) \mid x_\phi(\rho,\sigma) = 1_F\} = \sum_{\sigma\in\phi(\rho)} y(\phi)$. $\square$

Theorem 58 (Representation)**.** If K is a KAT and F is the KAT of all binary relations on a finite set S, then $K \boxplus F \cong \mathsf{Mat}(S,K)$.

Proof. Recall that the KAT F of binary relations on a finite set S is isomorphic to $\mathsf{Mat}(S,2_K)$ where $2_K = \{0_K, 1_K\}$, and hence it is a subalgebra of $\mathsf{Mat}(S,K)$. Define the map

$$h : \mathsf{Trm}(K,F) \to \mathsf{Mat}(S,K)$$

by putting $h(c_x) = x1_{\mathsf{Mat}(S,K)}$ (where the operation here is scalar multiplication) and $h(c_\phi) = \phi$. Note that for every equation $f = g$ in $\Delta_K \cup \Delta_F$ it holds that $h(f) = h(g)$ in $\mathsf{Mat}(S,K)$. For example, if $x,y \in K$ then the equation $c_x; c_y = c_{x;y}$ is in the diagram Δ_K and we have

$$h(c_x;c_y) = h(c_x)\cdot h(c_y) = (x1_{\mathsf{Mat}(S,K)})\cdot(y1_{\mathsf{Mat}(S,K)}) = (x;y)1_{\mathsf{Mat}(S,K)} = h(c_{x;y}).$$

For a commutativity equation $c_x; c_\phi = c_\phi; c_x$ in D observe that:

$$h(c_x;c_\phi) = h(c_x)\cdot h(c_\phi) = (x1_{\mathsf{Mat}(S,K)})\cdot\phi = x\phi$$
$$h(c_\phi;c_x) = h(c_\phi)\cdot h(c_x) = \phi\cdot(x1_{\mathsf{Mat}(S,K)}) = x\phi$$

Since $\mathsf{Mat}(S,K)$ is a KAT, it follows that for every equation $f = g$ in $E_{K,F}$ we have $h(f) = h(g)$ in $\mathsf{Mat}(S,K)$. So, we can define the homomorphism $h_1 : K \boxplus F \to \mathsf{Mat}(S,K)$ on the equivalence classes. We claim that h_1 is surjective. Indeed, we have for every matrix M in $\mathsf{Mat}(S,K)$:

$$h_1\left(\textstyle\sum_{\rho,\sigma}\iota_1(M_{\rho\sigma});\iota_2(\rho?;\sigma!)\right) = \textstyle\sum_{\rho,\sigma} M_{\rho\sigma}(\rho?;\sigma!) = M.$$

It remains to show that h_1 is injective. Let f, g be terms of $\mathsf{Trm}(K,F)$ with $h_1([f]) = h_1([g])$. We know from Lemma 57 that there are functions $x,y : S\times S \to K$ such that:

$$[f] = \textstyle\sum_{\rho,\sigma}\iota_1(x(\rho,\sigma));\iota_2(\rho?;\sigma!) \qquad [g] = \textstyle\sum_{\rho,\sigma}\iota_1(y(\rho,\sigma));\iota_2(\rho?;\sigma!)$$

The function x and y are, in fact, elements of $\mathsf{Mat}(S, K)$. From the assumption we get:

$$\begin{aligned} h_1([f]) = h_1([g]) &\implies \textstyle\sum_{\rho,\sigma} x(\rho,\sigma)(\rho?;\sigma!) = \sum_{\rho,\sigma} y(\rho,\sigma)(\rho?;\sigma!) \\ &\implies x(\rho,\sigma) = y(\rho,\sigma) \text{ for all } \rho,\sigma \\ &\implies x = y. \end{aligned}$$

It follows that $[f] = [g]$, and the proof is thus complete. $\square$

Corollary 59. If K is a KAT and F is any KAT of binary relations on a finite set S, then $K \boxplus F$ is isomorphic to a subalgebra of $\mathsf{Mat}(S, K)$.

Proof. Let F' be the KAT of all binary relations on S, which means that there is an injective homomorphism $k : F \to F'$. This lifts to an injective homomorphism $k : K \boxplus F \to K \boxplus F'$. Since $K \boxplus F' \cong \mathsf{Mat}(S, K)$ by Theorem 58, $K \boxplus F$ is isomorphic to a subalgebra of $\mathsf{Mat}(S, K)$. $\square$

Corollary 60 (Injectivity)**.** Let K be an arbitrary KAT and F be a finite KAT. Then, the coprojection map $\iota_1 : K \to K \boxplus F$ is injective, i.e., $\iota_1(x) = \iota_1(y)$ implies that $x = y$.

Proof. The isomorphism $h_1 : K \boxplus F \to \mathsf{Mat}(F, K)$ that we defined in the proof of Theorem 58 sends $\iota_1(x)$ to the matrix $x1_{\mathsf{Mat}(F,K)}$. So, the assumption $\iota_1(x) = \iota_1(y)$ implies that $h_1(\iota_1(x)) = h_1(\iota_1(y))$, and therefore $x = y$. $\square$

If we instantiate the finite KAT F of Corollary 60 to the free KAT K_n over the mutable tests T_n (see Theorem 53), then we obtain that every KAT K can be conservatively extended with n mutable tests. So, this extension $K \boxplus K_n$ does not affect the theory of K.

3.4 KAT and Extra Mutable Tests

Using the commutative coproduct construction of the previous section and the crucial representation result (Theorem 58), we will obtain easily several concrete completeness theorems.

Let $(K, B, +, ;, ^*, 0, 1, \neg)$ be an arbitrary KAT. We consider expressions that involve constants for the elements of K as well as extra mutable tests T_n. The grammar is:

$$\text{test terms } p, q ::= 0 \mid 1 \mid c_b \text{ for } b \in B \mid t? \mid \bar{t}? \mid p + q \mid p; q \mid \neg p$$

$$\text{terms } f, g ::= \text{test term } p \mid c_x \text{ for } x \in K \mid f + g \mid f; g \mid f^*$$

We write $\mathsf{Trm}(K, T_n)$ for the set of all these terms.

Claim 61 (Commutativity Equations)**.** For every element $x \in K$ and every term $f \in \mathsf{Trm}(T_n)$, the equation $c_x; f = f; c_x$ is provable in $\mathsf{KAT}+D_n$, where D_n consists of the equations

$$c_x; t? = t?; c_x \qquad c_x; \bar{t}? = \bar{t}?; c_x \qquad c_x; t! = t!; c_x \qquad c_x; \bar{t}! = \bar{t}!; c_x$$

for every $x \in K$ and $t \in T_n$.

Proof. The base cases for $t?$, $\bar{t}?$, $t!$, and $\bar{t}!$ are immediate from D_n. Moreover, $c_x; 1 = c_x = c_x; 1$ and $c_x; 0 = 0 = 0; c_x$ for the constants 1 and 0 respectively. For the step cases of choice and composition, we see (using the induction hypothesis) that $c_x; f; g = f; c_x; g = f; g; c_x$ and

$$c_x; (f + g) = c_x; f + c_x; g = f; c_x + g; c_x = (f + g); c_x.$$

Finally, for iteration we observe that $c_x; f^* = f^*; c_x$ is implied by $c_x; f = f; c_x$ (bisimulation rule 1.3), which is provable by the induction hypothesis. $\square$

Theorem 62 (Completeness)**.** The axioms $\mathsf{KAT}+\mathsf{Mut}_n+\Delta_K+D_n$ are complete for the equational theory of $K \boxplus K_n$, where K_n is the free KAT over the mutable tests T_n.

Proof. Theorem 53 says that we can assume without loss of generality that the algebra K_n is the KAT of all binary relations on the set S of T_n-assignments $T_n \to 2$ (see Definition 52). The term translation function $[\,\cdot\,] : \mathsf{Trm}(K, T_n) \to \mathsf{Trm}(K, K_n)$ is defined as:

$$[c_x] = c_x \text{ for } x \in K \qquad [t?] = c_{(t?)_S} \qquad [\bar{t}?] = c_{(\bar{t}?)_S} \qquad [t!] = c_{(t!)_S} \qquad [\bar{t}!] = c_{(\bar{t}!)_S}$$

The canonical interpretation $H : \mathsf{Trm}(K, K_n) \to K \boxplus K_n$ is specified by $H(c_x) = \iota_1(x)$ for $x \in K$ and $H(c_\phi) = \iota_2(\phi)$ for $\phi \in K_n$. We will establish completeness with respect to the interpretation

$$H \circ [\,\cdot\,] : \mathsf{Trm}(K, T_n) \to K \boxplus K_n.$$

Let f and g be terms in $\mathsf{Trm}(K, T_n)$ for which it holds that $H([f]) = H([g])$. By definition of $K \boxplus K_n$ this implies that the equation $[f] = [g]$ is provable in $\mathsf{KAT}+\Delta_K+\Delta_{K_n}+D$, where D consists of the equations $c_x; c_\phi = c_\phi; c_x$ for $x \in K$ and $\phi \in K_n$.

We know that there is a proof in the system $\mathsf{KAT}+\Delta_K+\Delta_{K_n}+D$ for the equation $[f] = [g]$. We replace in the proof every symbol c_ϕ with the term $\sum_{\sigma\in\phi(\rho)} \mathsf{test}(\rho); \mathsf{set}(\sigma)$. Now, the resulting proof is actually a proof in the system $\mathsf{KAT}+\mathsf{Mut}_n+\Delta_K+D_n$. This is because every equation $c_x; c_\phi = c_\phi; c_x$ of D corresponds to a provable equation of the form $c_x; f = f; c_x$ given by Claim 61. Moreover, every equation of the diagram Δ_{K_n} can certainly be simulated in the system $\mathsf{KAT}+\mathsf{Mut}_n$, because it is complete for the equational theory of K_n. $\square$

Corollary 63 (Completeness)**.** Let B and Σ be finite alphabets of primitive tests and actions respectively. We write K for the free KAT on generators B and Σ.

The axioms $\mathsf{KAT}+\mathsf{Mut}_n+D''$, where D'' consists of the equations

$$p; t! = t!; p \qquad p; \bar{t}! = \bar{t}!; p \qquad \neg p; t! = t!; \neg p \qquad \neg p; \bar{t}! = \bar{t}!; \neg p$$

$$a; t? = t?; a \qquad a; \bar{t}? = \bar{t}?; a \qquad a; t! = t!; a \qquad a; \bar{t}! = \bar{t}!; a$$

for $p \in B$, $a \in \Sigma$ and $t \in T_n$, are complete for the equational theory of $K \boxplus K_n$.

Proof. The axioms of D'' are sufficient to prove the equations

$$f; t? = t?; f \qquad f; \bar{t}? = \bar{t}?; f \qquad f; t! = t!; f \qquad f; \bar{t}! = \bar{t}!; f$$

for every term over B and Σ. Theorem 62 then gives us easily the result. $\square$

3.4.1 A Folk Theorem of Program Schematology

In this section we illustrate how the system $\mathsf{KAT}+\mathsf{Mut}+D$ can be used in practice. We will show, reasoning equationally in $\mathsf{KAT}+\mathsf{Mut}+D$, a classical result of program schematology: Every while program can be simulated by a while program with at most one while loop, assuming that we allow extra Boolean variables.

We work with an imperative programming language that has atomic programs Σ (written $a, b, \ldots$), the constant program skip, atomic tests B, as well as the constructs: sequential composition $f; g$, conditional $\mathsf{if}\, p\, \mathsf{then}\, f\, \mathsf{else}\, g$, and iteration $\mathsf{while}\, p\, \mathsf{do}\, f$. These programming constructs are modeled in KAT as follows:

$$\mathsf{skip} = 1 \qquad \mathsf{if}\, e\, \mathsf{then}\, f\, \mathsf{else}\, g = ef + \bar{e}g$$

$$f; g = fg \qquad \mathsf{while}\, e\, \mathsf{do}\, f = (ef)^*\bar{e}$$

We omit the sequential composition symbol ; in our KAT terms to reduce the notational clutter, and we write $\bar{e}$ to mean $\neg e$. There is a semantic justification for these translations, using the standard relation-theoretic semantics for the input-output behavior of while programs.

The intuition for the construction we will present is that we need to introduce extra Boolean variables that encode the control structure of the program. These variables are modeled in $\mathsf{KAT}+\mathsf{Mut}+D$ using extra mutable tests $t_1, t_2, t_3, \ldots$, which are disjoint from B and Σ.

Commutativity axioms: The collection D of axioms, which forms part of the system $\mathsf{KAT}+\mathsf{Mut}+D$, includes the following equations:

$$\begin{array}{cccc} pt! = t!p & p\bar{t}! = \bar{t}!p & \bar{p}t! = t!\bar{p} & \bar{p}\bar{t}! = \bar{t}!\bar{p} \\ at? = t?a & a\bar{t}? = \bar{t}?a & at! = t!a & a\bar{t}! = \bar{t}!a \end{array}$$

for every primitive test $p \in B$ and every atomic program $a \in \Sigma$. The above axioms say that the assignments $t!$ and $\bar{t}!$ do not affect the truth value of primitive regular tests. Moreover, a primitive program $a \in \Sigma$ does not affect the truth value of mutable tests.

Claim 64 (Test Commutativity)**.** Let p be a test term over B and T (i.e., one that may involve both regular and mutable tests). If the mutable test symbols $t?$ and $\bar{t}?$ do not appear in p, then the commutativity equations $t!p = pt!$ and $\bar{t}!p = p\bar{t}!$ are provable in $\mathsf{KAT}+\mathsf{Mut}+D$.

Proof. Negations in tests can be pushed down to the leaves, so we assume without loss of generality that only primitive tests $p \in B$ can be negated. The proof proceeds by induction on the test term. For an atomic test p or $\bar{p}$ with $p \in B$, the claim follows directly from the axioms of D. The cases of the constants 0 and 1 are trivial. For a mutable test $s?$, our assumption says that $s \neq t$ and therefore $t!s? = s?t!$ and $\bar{t}!s? = s?\bar{t}!$ are axioms of Mut. The argument is analogous for a mutable test $\bar{s}?$. For the induction step, consider the case $p+q$ of choice:

$$t!(p+q) = t!p + t!q = pt! + qt! = (p+q)t!$$

$$\bar{t}!(p+q) = \bar{t}!p + \bar{t}!q = p\bar{t}! + q\bar{t}! = (p+q)\bar{t}!$$

The case pq is equally easy: $t!pq = pt!q = pqt!$ and $\bar{t}!pq = p\bar{t}!q = pq\bar{t}!$. $\square$

Claim 65 (Commutativity)**.** Let f be an arbitrary term over B, Σ and T. If the mutable test symbols $t, \bar{t}$ do not appear in f, then the following equations are provable in KAT+Mut+D:

$$t?f = ft? \qquad \bar{t}?f = f\bar{t}? \qquad t!f = ft! \qquad \bar{t}!f = f\bar{t}!$$

Proof. We only deal with the equation $t!f = ft!$, because for the other equations the proof is completely analogous. We argue by induction on the structure of the term f. If the term is a test, then the result follows from Claim 64. For an atomic program $a \in \Sigma$, the stipulated axioms in D gives us the equation $t!a = at!$. For an assignment $s!$ the hypothesis says that $s \neq t$ and therefore $t!s! = s!t!$ from Mut. The argument is similar for an assignment $\bar{s}!$. For composition and choice we have using the induction hypothesis:

$$t!fg = ft!g = fgt! \qquad t!(f+g) = t!f + t!g = ft! + gt! = (f+g)t!.$$

It remains to derive the equation $t!f^* = f^*t!$. By virtue of the bisimulation rule (1.3), it suffices to see that $t!f = ft!$, which is the induction hypothesis. $\square$

The main theorem of this section (Theorem 69 below) is a normal form theorem, from which the result we want to show follows immediately. Working in a bottom-up fashion, every while program term is brought in the normal form. That the transformed program in normal form is equivalent to the original one is shown in KAT+Mut+D.

A *normal form* is a term $u; \mathsf{while}\, p\, \mathsf{do}\, \phi; z$, where u and ϕ are while-free terms over Σ, B and T, p is a test-term over B and T, and z is of the form $\bar{t}_1!\bar{t}_2! \cdots \bar{t}_k!$. So, the pre-computation u, the while-guard p, and the while-body ϕ may involve any of the extra mutable test symbols $t_1, \ldots, t_k, \bar{t}_1, \ldots, \bar{t}_k$. The post-computation $z = \bar{t}_1!\bar{t}_2! \cdots \bar{t}_k!$ "zeroes out" all the extra mutable Boolean variables. Its role is in some sense to simply project out the extra finite state that was used to model control-flow.

Claim 66 (Base Case)**.** Every while-free program f over B, Σ can be brought to normal form.

Proof. Suppose that f is a while-free program term, and let t be a fresh mutable test symbol. Intuitively, $t?$ holds if f has not been executed yet, and $\bar{t}?$ holds after f has been executed. Reasoning in KAT+Mut+D we will prove the equation

$$f; z = t!; \mathsf{while}\, t?\, \mathsf{do}\, (f; \bar{t}!); z, \text{ where } z = \bar{t}!.$$

The main idea is to unfold the expression $(t?f\bar{t}!)^*$ twice to obtain:

$$\begin{aligned}
(t?f\bar{t}!)^* &= 1 + t?f\bar{t}!(t?f\bar{t}!)^* && [\text{unfold } ^*] \\
&= 1 + t?f\bar{t}!(1 + t?f\bar{t}!(t?f\bar{t}!)^*) && [\text{unfold } ^*] \\
&= 1 + t?f\bar{t}! + t?f\bar{t}!t?f\bar{t}!(t?f\bar{t}!)^* && [\text{distributivity}] \\
&= 1 + t?f\bar{t}!. && [\text{because } \bar{t}!t? = 0]
\end{aligned}$$

We have used above the property $\bar{t}!t? = \bar{t}!\bar{t}?t? = \bar{t}!0 = 0$. We tranform the RHS of the equation:

$$\begin{aligned}
\text{RHS} &= t!(t?f\bar{t}!)^*\neg t?\bar{t}! && [\text{encoding}] \\
&= t!(t?f\bar{t}!)^*\bar{t}? && [\text{equations } \neg t?\bar{t}! = \bar{t}?\bar{t}! = \bar{t}?] \\
&= t!(1 + t?f\bar{t}!)\bar{t}? && [\text{see above}] \\
&= t!\bar{t}? + t!t?f\bar{t}!\bar{t}? && [\text{distributivity}]
\end{aligned}$$

$$
\begin{aligned}
&= t!t?f\bar{t}!\bar{t}?\bar{t}! && [\text{equation } t!\bar{t}? = 0]\\
&= t!f\bar{t}!\bar{t}! && [\text{equations } t!t? = t! \text{ and } \bar{t}!\bar{t}? = \bar{t}!]\\
&= t!f\bar{t}!. && [\text{equation } \bar{t}!\bar{t}! = \bar{t}!]
\end{aligned}
$$

Since the symbols t and $\bar{t}$ do not appear in the term f, we can derive by virtue of Claim 65 the equations $t!f\bar{t}! = ft!\bar{t}! = f\bar{t}! = f; z$. □

Claim 67 (Conditional)**.** Let f and g be while programs over Σ, B. Suppose that the equations

$$f; z = u; \mathsf{while}\, p\, \mathsf{do}\, \phi; z \qquad\qquad g; z = u; \mathsf{while}\, q\, \mathsf{do}\, \psi; z$$

are provable (where the right-hand side of each equation above is a normal form). Then,

$$
\begin{aligned}
(\mathsf{if}\, e\, \mathsf{then}\, f\, \mathsf{else}\, g); z; \bar{t}! \;=\; & \mathsf{if}\, e\, \mathsf{then}\, (t!; u)\, \mathsf{else}\, (\bar{t}!; v);\\
& \mathsf{while}\, ((t? \wedge p) \vee (\bar{t}? \wedge q))\, \mathsf{do}\, (\mathsf{if}\, t?\, \mathsf{then}\, \phi\, \mathsf{else}\, \psi);\\
& z; \bar{t}!
\end{aligned}
$$

is provable, where t is a fresh symbol for a mutable test.

Remark. The intuition for the given translation is that the fresh variable t records the branch of the conditional that should be taken. So, $t?$ holds when f should be executed, and $\bar{t}?$ holds when g should be executed.

Proof. The while-free pre-computation in the normal form translation is equal to $et!u + \bar{e}\bar{t}!v$. The guard of the while loop is $t?p + \bar{t}?q$, and the body is $t?\phi + \bar{t}?\psi$. So,

$$
\begin{aligned}
((t? \wedge p) \vee (\bar{t}? \wedge q)); (\mathsf{if}\, t?\, \mathsf{then}\, \phi\, \mathsf{else}\, \psi) &= (t?p + \bar{t}q)(t?\phi + \bar{t}?\psi)\\
&= t?pt?\phi + t?p\bar{t}?\psi + \bar{t}qt?\phi + \bar{t}q\bar{t}?\psi\\
&= t?p\phi + \bar{t}?q\psi.
\end{aligned}
$$

The negation of the guard of the loop is

$$\neg(t?p + \bar{t}?q) = (\bar{t}? + \bar{p})(t? + \bar{q}) = \bar{t}?\bar{q} + t?\bar{p} + \bar{p}\bar{q}.$$

First, we claim that $t?(t?p\phi)^* = t?(p\phi)^*$. Since $t? \leq 1$ and * is monotone, we have that $(t?p\phi)^* \leq (p\phi)^*$, and therefore $t?(t?p\phi)^* \leq t?(p\phi)^*$. In order to show that $t?(p\phi)^* \leq t?(t?p\phi)^*$, it suffices to see that $t? \leq t?(t?p\phi)^*$, and that

$$\begin{aligned}
t?(t?p\phi)^*p\phi &= t?(1 + (t?p\phi)^*t?p\phi)p\phi && \text{[unfold } ^*\text{]} \\
&= t?p\phi + t?(t?p\phi)^*t?p\phi p\phi && \text{[distributivity]} \\
&= t?t?p\phi + t?(t?p\phi)^*t?t?p\phi p\phi && \text{[equation } t?t? = t?\text{]} \\
&= t?t?p\phi + t?(t?p\phi)^*t?p\phi t?p\phi && [t \text{ not in } p, \phi] \\
&= t?(1 + (t?p\phi)^*t?p\phi)t?p\phi && \text{[distributivity]} \\
&= t?(t?p\phi)^*t?p\phi && \text{[fold } ^*\text{]} \\
&\leq t?(t?p\phi)^*. && \text{[inequality } x^*x \leq x^*\text{]}
\end{aligned}$$

Now, we want to show that $t?(t?p\phi + \bar{t}?q\psi)^* = t?(t?p\phi)^*$. By monotonicity of *, the right-hand side is less than or equal to the left-hand side. For the other part, we need to show that

$$\begin{aligned}
&t?(t?p\phi)^*(t?p\phi + \bar{t}?q\psi) = && \text{[equation } t?t? = t?\text{]} \\
&t?t?(t?p\phi)^*(t?p\phi + \bar{t}?q\psi) = && \text{[previous claim]} \\
&t?t?(p\phi)^*(t?p\phi + \bar{t}?q\psi) = && [t \text{ not in } p, \phi] \\
&t?(p\phi)^*t?(t?p\phi + \bar{t}?q\psi) = && \text{[distributivity and } t?\bar{t}? = 0\text{]} \\
&t?(p\phi)^*t?p\phi = && \text{[previous claim]} \\
&t?(t?p\phi)^*t?p\phi,
\end{aligned}$$

which is $\leq t?(t?p\phi)^*$. Let W abbreviate the entire while loop of the normal form translation. We have already seen that $W = (t?p\phi + \bar{t}?q\psi)^*(\bar{t}?\bar{q} + t?\bar{p} + \bar{p}\bar{q})$ and therefore

$$\begin{aligned} t?W &= t?(t?p\phi)^*(\bar{t}?\bar{q} + t?\bar{p} + \bar{p}\bar{q}) && \text{[previous claim]} \\ &= t?(p\phi)^*(\bar{t}?\bar{q} + t?\bar{p} + \bar{p}\bar{q}) && \text{[previous claim]} \\ &= (p\phi)^* t?(\bar{t}?\bar{q} + t?\bar{p} + \bar{p}\bar{q}) && [t \text{ not in } p, \phi] \\ &= (p\phi)^*(t?\bar{t}?\bar{q} + t?t?\bar{p} + t?\bar{p}\bar{q}) && \text{[distributivity]} \\ &= (p\phi)^*(t?\bar{p} + t?\bar{p}\bar{q}) && [t?\bar{t}? = 0 \text{ and } t?t? = t?] \\ &= (p\phi)^* t?\bar{p}. && [\text{because } t?\bar{p}\bar{q} \leq t?\bar{p}] \end{aligned}$$

We denote by RHS the right-hand side of the equation we want to prove, and we observe:

$$\begin{aligned} e\text{RHS} &= e(et!u + \bar{e}\bar{t}!v)Wz\bar{t}! && \text{[encoding]} \\ &= et!uWz\bar{t}! && [ee = e \text{ and } e\bar{e} = 0] \\ &= et!t?uWz\bar{t}! && [\text{equation } t!t? = t!] \\ &= et!ut?Wz\bar{t}! && [t \text{ does not appear in } u] \\ &= et!u(p\phi)^* t?\bar{p}z\bar{t}! && \text{[previous claim]} \\ &= eu(p\phi)^*\bar{p}z\bar{t}!, && [t \text{ not in } u, p, \phi, z \text{ and } t!t?\bar{t}! = t!\bar{t}! = \bar{t}!] \end{aligned}$$

which is equal to $efz\bar{t}!$ by the induction hypothesis. Similarly, the equation $\bar{e}\text{RHS} = \bar{e}gz\bar{t}!$ can be derived. We thus conclude that

$$\text{RHS} = (e + \bar{e})\text{RHS} = e\text{RHS} + \bar{e}\text{RHS} = efz\bar{t}! + \bar{e}gz\bar{t}! = (ef + \bar{e}g)z\bar{t}!,$$

which is equal to $(\mathsf{if}\, e \,\mathsf{then}\, f \,\mathsf{else}\, g); z; \bar{t}!$, namely the left-hand size of the desired equation. □

Claim 68 (Composition)**.** Let f and g be while programs over Σ, B. Suppose that the equations

$$f; z = u; \mathsf{while}\, p \,\mathsf{do}\, \phi; z \qquad\qquad g; z = u; \mathsf{while}\, q \,\mathsf{do}\, \psi; z$$

are provable (where the right-hand side of each equation above is a normal form). Then, the equation

$$\begin{aligned} f; g; z; \bar{t}! \;=\; & t!; u; \\ & \textsf{while}\,(t? \vee (\bar{t}? \wedge q))\,\textsf{do} \\ & \quad \textsf{if}\; t?\;\textsf{then}\;(\textsf{if}\; p\;\textsf{then}\;\phi\;\textsf{else}\;(z; \bar{t}!; v))\;\textsf{else}\;\psi; \\ & z; \bar{t}!. \end{aligned}$$

is provable, where t is a fresh symbol for a mutable test.

Remark. The idea for the translation is that the fresh variable t records the position of the execution. That is, $t?$ holds while f is being executed, and $\bar{t}?$ holds while g is being executed.

Proof. The negation of the guard of the while loop is $\neg(t?+\bar{t}?q) = \bar{t}?(t?+\bar{q}) = \bar{t}?\bar{q}$. The body of the loop is equal to $t?(p\phi + \bar{p}z\bar{t}!v) + \bar{t}?\psi = t?p\phi + t?\bar{p}z\bar{t}!v + \bar{t}?\psi$. So, the Fisher-Ladner encoding of the while loop is

$$\begin{aligned} &[(t? + \bar{t}?q)(t?p\phi + t?\bar{p}z\bar{t}!v + \bar{t}?\psi)]^*\bar{t}?\bar{q} = && \text{[distributivity]} \\ &[t?p\phi + t?\bar{p}z\bar{t}!v + \bar{t}?q\psi]^*\bar{t}?\bar{q} = && \text{[abbreviation]} \\ &(A + \bar{t}?q\psi)^*\bar{t}?\bar{q} = && \text{[denesting rule (1.2)]} \\ &A^*(\bar{t}?q\psi A^*)^*\bar{t}?\bar{q}, \end{aligned}$$

where we put $A = t?p\phi + t?\bar{p}z\bar{t}!v$.

From $\bar{t}?A = \bar{t}?(t?p\phi + t?\bar{p}z\bar{t}!v) = 0 \leq \bar{t}?$ we obtain that $\bar{t}?A^* \leq \bar{t}?$. Moreover, $\bar{t}? \leq \bar{t}?A^*$ and hence $\bar{t}?A^* = \bar{t}?$. It follows that $\bar{t}?q\psi A^* = q\psi\bar{t}?A^* = q\psi\bar{t}?$. Now, we claim that $(q\psi\bar{t}?)^*\bar{t}? = \bar{t}?(q\psi)^*$. The inequality $(q\psi\bar{t}?)^*\bar{t}? \leq \bar{t}?(q\psi)^*$ follows from monotonicity of *. For the inequality $\bar{t}?(q\psi)^* \leq (q\psi\bar{t}?)^*\bar{t}?$ we need to show that

$$(q\psi\bar{t}?)^*\bar{t}?q\psi = (q\psi\bar{t}?)^*q\psi\bar{t}? = (q\psi\bar{t}?)^*q\psi\bar{t}?\bar{t}? \leq (q\psi\bar{t}?)^*\bar{t}?$$

We have thus shown that the while loop is equal to $A^*(q\psi\bar{t}?)^*\bar{t}?q = A^*\bar{t}?(q\psi)^*\bar{q}$.

Now, we focus on simplifying the expression $t?A^*\bar{t}? = t?(t?p\phi + t?\bar{p}z\bar{t}!v)^*\bar{t}?$. First, we observe that unfolding $(t?\bar{p}z\bar{t}!v)^*$ twice gives us the equation

$$(t?\bar{p}z\bar{t}!v)^* = 1 + t?\bar{p}z\bar{t}!v. \tag{3.3}$$

Moreover, $\bar{t}?(t?p\phi)^* = \bar{t}?(1 + t?p\phi(t?p\phi)^*) = \bar{t}?$. Therefore, using the denesting rule, we obtain

$$\begin{aligned}
t?A^*\bar{t}? &= t?(t?p\phi + t?\bar{p}z\bar{t}!v)^*\bar{t}? && \text{[definition of } A\text{]}\\
&= t?(t?p\phi)^*(t?\bar{p}z\bar{t}!v(t?p\phi)^*)^*\bar{t}? && \text{[denesting rule]}\\
&= t?(t?p\phi)^*(t?\bar{p}z\bar{t}!v\bar{t}?(t?p\phi)^*)^*\bar{t}? && [\bar{t}! = \bar{t}!\bar{t}? \text{ and } t \text{ not in } v]\\
&= t?(t?p\phi)^*(t?\bar{p}z\bar{t}!v\bar{t}?)^*\bar{t}? && \text{[claim above]}\\
&= t?(t?p\phi)^*(t?\bar{p}z\bar{t}!v)^*\bar{t}? && [t \text{ not in } v \text{ and } \bar{t}! = \bar{t}!\bar{t}?]\\
&= t?(t?p\phi)^*(1 + t?\bar{p}z\bar{t}!v)\bar{t}? && \text{[equation 3.3]}\\
&= t?(t?p\phi)^*\bar{t}? + t?(t?p\phi)^*t?\bar{p}z\bar{t}!v\bar{t}? && \text{[distributivity]}\\
&= t?(t?p\phi)^*t?\bar{p}z\bar{t}!v\bar{t}? && \text{[first term 0]}\\
&= t?(p\phi)^*\bar{p}z\bar{t}!v. && [\text{because } t?(t?p\phi)^*t? = t?(p\phi)^*]
\end{aligned}$$

Finally, we can work on the right-hand side of the equation we want to establish:

$$\begin{aligned}
\text{RHS} &= t!uA^*\bar{t}?(q\psi)^*\bar{q}z\bar{t}! && [\text{while loop is } A^*\bar{t}?(q\psi)^*\bar{q}]\\
&= t!ut?A^*\bar{t}?(q\psi)^*\bar{q}z\bar{t}! && [t! = t!t? \text{ and } t \text{ not in } u]\\
&= t!ut?(p\phi)^*\bar{p}z\bar{t}!v(q\psi)^*\bar{q}z\bar{t}! && \text{[claim above]}\\
&= u(p\phi)^*\bar{p}zv(q\psi)^*\bar{q}z\bar{t}!, && [t \text{ not in } u, p, \phi, z, v, q, \psi]
\end{aligned}$$

which is equal by the assumptions to $fzgz\bar{t}! = fgzz\bar{t}! = f; g; z; \bar{t}!$. □

Theorem 69 (The Folk Theorem)**.** For every while program f over Σ, B, there are while-free u, p, ϕ and a finite collection $t_1, \ldots, t_k$ of extra mutable tests such that $f; z = u; \mathsf{while}\, p \,\mathsf{do}\, \phi; z$ is provable in the system KAT+Mut+D, where $z = \bar{t}_1!; \ldots; \bar{t}_k!$.

Proof. The proof is by induction on the while program. Claim 66, Claim 67, and Claim 68 already cover all the base cases, as well as the inductive cases of conditionals and sequential composition. It remains to consider the case of while loops. So, we suppose that

$$f; z = u; \mathsf{while}\, p \,\mathsf{do}\, \phi; z \quad \text{(equivalently, } fz = u(p\phi)^* \bar{p} z\text{)}$$

is provable, where the right-hand side of the equation is a normal form. First, we derive

$$(\mathsf{while}\, e \,\mathsf{do}\, f); z = (\mathsf{while}\, e \,\mathsf{do}\, (f; z)); z.$$

The left-hand side is equal to $(ef)^* \bar{e} z$, and the right-hand side equal to $(efz)^* \bar{e} z$. Since the test $\bar{e}$ contains no mutable symbols, it suffices to show that $(ef)^* z = (efz)^* z$. Now,

$$(efz)^* z \leq (ef)^* z \Longleftarrow efz(ef)^* z \leq (ef)^* z,$$

which holds because $efz(ef)^* z = ef(ef)^* zz \leq (ef)^* z$. Now, we observe that $(efz)^* z = z(efz)^*$ by the bisimulation rule (1.3), because $(efz)z = z(efz)$ (both are equal to efz). So,

$$(ef)^* z \leq (efz)^* z \Longleftarrow ef(efz)^* z \leq (efz)^* z,$$

which holds because $ef(efz)^* z = ef(efz)^* zz = efz(efz)^* z \leq (efz)^* z$. Using the claim we have just proved, we can bring the program in a more convenient form:

$$(\mathsf{while}\, e \,\mathsf{do}\, f); z = \Big(\mathsf{if}\, e \,\mathsf{then}\, \big(u; \mathsf{while}\, (e + p) \,\mathsf{do}\, (\mathsf{if}\, p \,\mathsf{then}\, \phi \,\mathsf{else}\, (z; u))\big)\Big); z.$$

The right-hand side of the above equation is provably equal to

$$\begin{aligned}\bar{e}z + eu[(e+p)(p\phi + \bar{p}zu)]^*(\overline{e+p})z &= \bar{e}z + eu[ep\phi + e\bar{p}zu + p\phi]^*\bar{e}\bar{p}z \\ &= \bar{e}z + eu(p\phi + e\bar{p}zu)^*\bar{e}\bar{p}z,\end{aligned}$$

because $ep\phi \leq p\phi$. Using the denesting rule $(x+y)^* = x^*(yx^*)^*$ and the sliding rule $(xy)^*x = x(yx)^*$, we see that this term is equal to

$$\begin{aligned}\bar{e}z + eu(p\phi + e\bar{p}zu)^*\bar{e}\bar{p}z &= \bar{e}z + eu(p\phi)^*(e\bar{p}zu(p\phi)^*)^*\bar{e}\bar{p}z \\ &= \bar{e}z + eu(p\phi)^*(\bar{p}zeu(p\phi)^*)^*\bar{e}\bar{p}z \\ &= \bar{e}z + (eu(p\phi)^*\bar{p}z)^*eu(p\phi)^*\bar{e}\bar{p}zz \\ &= \bar{e}z + (efz)^*eu(p\phi)^*\bar{p}z\bar{e}z \\ &= \bar{e}z + (efz)^*(efz)\bar{e}z \\ &= (1 + (efz)^*(efz))\bar{e}z,\end{aligned}$$

which is equal to $(efz)^*\bar{e}z = (\mathsf{while}\, e\, \mathsf{do}\, (f; z)); z = (\mathsf{while}\, e\, \mathsf{do}\, f); z$. But we already know how to deal with conditional statements, so we use Claim 67 and we are done. □

3.5 Conclusion

We have shown how to axiomatically extend KAT with a finite amount of additional mutable state. This extra feature allows certain program transformations to be effected at the propositional level without passing to a full first-order system. The extension is conservative and deductively complete relative to the theory of the underlying algebra. We have given a representation theorem of the free models in terms of matrices.

An intriguing open problem is whether the commutative coproduct of two

KATs is injective. We have shown that it is if one of the two cofactors is a KAT of binary relations on a finite set.

CHAPTER 4
KLEENE ALGEBRA AND THEORIES OF FIXPOINTS

4.1 Introduction

In the realm of equational systems for reasoning about iteration, two chief complementary bodies of work stand out. One of these is *iteration theories* (IT), the subject of the extensive monograph of Bloom and Ésik [17] as well as many other authors (see the cited literature). The primary motivation for iteration theories is to capture in abstract form the equational properties of iteration on structures that arise in domain theory and program semantics, such as continuous functions on ordered sets. Of central interest is the dagger operation †, a kind of parameterized least fixpoint operator, that when applied to an object representing a simultaneous system of equations gives an object representing the least solution of those equations. Much of the work on iteration theories involves axiomatizing or otherwise characterizing the equational theory of iteration as captured by †. Complete axiomatizations have been provided [42, 20, 34] as well as other algebraic and categorical characterizations [3, 4, 120].

Bloom and Ésik claim that "...the notion of an iteration theory seems to axiomatize the equational properties of all computationally interesting structures..." [19]. This is true to a certain extent, certainly if one is interested only in structures that arise in domain theory and programming language semantics. However, it is not the entire story.

Another approach to equational reasoning about iteration that has met with some success over the years is the notion of *Kleene algebra* (KA), the algebra of regular expressions. KA has a long history going back to the original paper of Kleene [59] and was further developed by Conway, who coined the name Kleene algebra in his 1971 monograph [31]. It has since been studied by many authors. KA relies on an iteration operator * that characterizes iteration in a different way from †. Its principal models are not those of domain theory, but rather basic algebraic objects such as sets of strings (in which * gives the Kleene asterate operation), binary relations (in which * gives reflexive transitive closure), and other structures with applications in shortest path algorithms on graphs and geometry of convex sets. Complete axiomatizations and complexity analyses have been given; the regular sets of strings over an alphabet A form the free KA on generators A in much the same way that the rational $\Sigma_\perp$-trees form the free IT on a signature Σ.

Although the two systems fulfill many of the same objectives and are related at some level, there are many technical and stylistic differences. Whereas iteration theories are based on Lawvere theories, a categorical concept, Kleene algebra operates primarily at a level of abstraction one click down. For this reason, KA may be somewhat more accessible. KA has been shown to be useful in several nontrivial static analysis and verification tasks (see e.g. [81, 62]). Also, KA can model nondeterministic computation, whereas IT is primarily deterministic.

Nevertheless, both systems have claimed to capture the notion of iteration in a fundamental way, and it is interesting to ask whether they can somehow be reconciled. This is the investigation that we undertake in this chapter. We start with the observation that ITs use the objects of a category to represent types. Technically the objects of interest in ITs are morphisms $f : n \to m$ in a category whose objects are natural numbers, and the morphism $f : n \to m$ is meant to model functions $f : A^m \to A^n$ (the arrows are reversed for technical reasons). Thus ITs might be captured by a version of KA with types. Although the primary version of KA is untyped, there is a notion of typed KA [74], although it only has types of the form $A \to B$, whereas to subsume IT it would need products as well. The presence of products allows ITs to capture parameterized fixpoints through the rule

$$\frac{f : n \to n + m}{f^\dagger : n \to m}$$

giving the parameterized least fixpoint $f^\dagger : A^m \to A^n$ of a parameterized function $f : A^m \times A^n \to A^n$. This would be possible to capture in KA if the typed version had products, which it does not. On the other hand, KA allows the modeling of nondeterministic computation, which IT does not, at least not in any obvious way. Thus to capture both systems, it would seem that we need to extend the type system of typed KA, or extend the categorical framework of IT to handle nondeterminism, or both.

The result of our investigation is a common categorical framework based on cartesian categories (categories with products) combined with a treatment of nondeterminism based on closure operators on the homsets. Types are represented by objects in the category, and we identify the appropriate axioms in the form of typed equations that allow equational reasoning on the morphisms. Our framework captures iteration as represented in ITs and KAs in a common language. We show how to define the KA operations as enrichments on the morphisms and how to define † in terms of *.

Our main contributions are as follows.

- **Nondeterminism from closure operators.** To accommodate nondeterminism, we consider a base category of "maps with possibly undefined/diverging components" and we represent the available nondeterministic choices with certain *closed sets* of morphisms. We axiomatize the relevant properties for these base categories and the closure operators, and we describe a general model-theoretic construction that gives rise to a *nondeterministic category* with an infinitary choice operation. The structure of binary products (pairing and projections) of the underlying category can be lifted in a smooth way to corresponding operations in the nondeterministic category.
- **Distinguishing the deterministic arrows**. Within our nondeterministic categories, certain properties work only for deterministic computations. We show how to capture the necessary properties of determinism by introducing a unary predicate D that distinguishes the deterministic elements of each homset. We axiomatize properties that concern the preservation of determinism, as well as properties of pairs that hold only in the deterministic part.
- **Continuity and Nondeterminism**. We specialize our construction of conservative extension with nondeterminism to the case where the base category is ω-continuous. This case is particularly relevant because such ω-continuous categories, which include standard domain-theoretic models, can be equipped with *least fixpoint operators*.
- **Capturing IT and KA**. The *equational theory* IT *of parametric fixpoints* is the set of equations between terms involving † that are true in the standard **CPO** model of ω-complete partial orders (ω-CPOs) and ω-continuous maps. Our goal is to capture this theory axiomatically using a typed vari-

ant of KA. For this, we consider the KA operations, including *, and a weakened axiomatization of nondeterministic categories that includes all the axioms of KA (except the strictness axiom). We show that our typed KA with products extends conservatively the theory IT. This is our main result.

- **Model theory**. Finally, our results imply that two particular constructions, the *lowerset-closure* construction on the category **Pposet** (of pointed posets and monotone maps) and the *ideal-closure* construction on the category **CPO** (of ω-CPOs and ω-continuous maps), provide natural concrete models in that they give rise to nondeterministic categories, and hence to typed KAs with products.

A detailed account of related work is given in §4.6.

4.2 Nondeterministic Structure from Closure Operators

As a first step in the development of our category-theoretic framework, we define axiomatically the class of *pointed ordered categories with products* (Def. 70 below), where the notion of undefinedness (or divergence) appears explicitly in the language as the constant $\bot$. The homsets of these categories are partially ordered by $\leq$, where the order is to be understood informally as follows: $f \leq g$ when f has more undefined components that g ("f is less defined than g").

The stipulated axioms capture properties of *non-strict pairs* (i.e., lazy or non-strict evaluation of pairs), which means intuitively that forming a pair $\langle v, \bot \rangle$ with an undefined element allows one to recover the other component. So,

product	$(X,Y) \mapsto X \times Y$	left projection	$\pi_1^{XY} : X \times Y \to X$
identity	$\mathsf{id}_X : X \to X$	right projection	$\pi_2^{XY} : X \times Y \to Y$
bottom	$\bot_{XY} : X \to Y$		

composition $$\frac{f : X \to Y \qquad g : Y \to Z}{f ; g : X \to Z}$$

pairing $$\frac{f : X \to Y \qquad g : X \to Z}{\langle f, g \rangle : X \to Y \times Z}$$

product $$\frac{f : X \to Y \qquad g : X' \to Y'}{f \times g \triangleq \langle \pi_1 ; f, \pi_2 ; g \rangle : X \times X' \to Y \times Y'}$$

Figure 4.1: Constants and operations for categories with binary products, where the homsets have additional pointed poset structure.

a non-strict pair with an undefined component is not itself necessarily undefined. Ordinarily, in the case of *eager pairs*, a pair $\langle v, \bot \rangle$ would be equal to $\bot$ by strictness. Intuitively, the computation of an eager pair $\langle v, \bot \rangle$, where v is a value and $\bot$ denotes a diverging computation, would also be diverging, i.e., $\langle v, \bot \rangle = \bot$. This makes it impossible to recover the left component v of the pair: $\langle v, \bot \rangle ; \pi_1 = \bot ; \pi_1 = \bot$.

Our goal in this section is to describe a general model-theoretic construction that enriches any such pointed ordered category $\mathcal{C}$ with additional *nondeterministic structure*. The idea is that we can model a nondeterministic program as a "closed" set of $\mathcal{C}$-arrows, where the elements of the set describe the available nondeterministic choices. For this we need to consider a family cl of *closure operators* for the homsets of $\mathcal{C}$ that interact reasonably with the category. Our main result is that the base category $\mathcal{C}$ and the closure operators cl give rise to a structure $\mathcal{C}_{\mathsf{cl}}$ that comes equipped with an *infinitary nondeterministic* operation. Moreover, there is an embedding from $\mathcal{C}$ into $\mathcal{C}_{\mathsf{cl}}$ and the image of this embedding is to be understood as the *deterministic subcategory* of $\mathcal{C}_{\mathsf{cl}}$.

A *typed algebraic structure* $\mathcal{C}$ consists of a class $\mathcal{C}_0$ of *objects*, a class $\mathcal{C}_1$ of *elements*, and the maps $\mathsf{dom}, \mathsf{cod} : \mathcal{C}_1 \to \mathcal{C}_0$ called *domain* and *codomain* respectively. For an element $f \in \mathcal{C}_1$ we write $f : X \to Y$ to denote that $\mathsf{dom}(f) = X$ and $\mathsf{cod}(f) = Y$. We then say that the expression $X \to Y$ is the *type* of f. The class $\mathcal{C}(X, Y)$, called a *homset*, consists of all elements of $\mathcal{C}$ that are of type $X \to Y$. In this paper we only consider typed structures whose homsets are sets, perhaps endowed with extra algebraic or order-theoretic structure.

Definition 70. A *pointed ordered category $\mathcal{C}$ with (binary) products* is a typed algebraic structure with the operations of Figure 4.1 and a partial order $\leq$ on every homset, that is a model of the following universal Horn axioms:

$$(f; g); h = f; (g; h) \text{ for } f : X \to Y,\ g : Y \to Z \text{ and } h : Z \to W \tag{4.1}$$

$$\mathsf{id}_X; f = f \text{ for } f : X \to Y \tag{4.2}$$

$$f; \mathsf{id}_Y = f \text{ for } f : X \to Y \tag{4.3}$$

$$\langle f, g\rangle; \pi_1^{YZ} = f \text{ for } f : X \to Y \text{ and } g : X \to Z \tag{4.4}$$

$$\langle f, g\rangle; \pi_2^{YZ} = g \text{ for } f : X \to Y \text{ and } g : X \to Z \tag{4.5}$$

$$\langle h; \pi_1^{YZ}, h; \pi_2^{YZ}\rangle = h \text{ for } h : X \to Y \times Z \tag{4.6}$$

$$f \leq f \text{ for } f : X \to Y \tag{4.7}$$

$$f \leq g \wedge g \leq h \implies f \leq h \text{ for } f, g, h : X \to Y \tag{4.8}$$

$$f \leq g \wedge g \leq f \implies f = g \text{ for } f, g, h : X \to Y \tag{4.9}$$

$$\bot_{XY} \leq f \text{ for } f : X \to Y \tag{4.10}$$

$$f \leq f' \wedge g \leq g' \implies f; g \leq f'; g' \text{ for } f, f' : X \to Y \text{ and } g, g' : Y \to Z \tag{4.11}$$

$$f \leq f' \wedge g \leq g' \implies \langle f, g\rangle \leq \langle f', g'\rangle \text{ for } f, f' : X \to Y \text{ and } g, g' : X \to Z \tag{4.12}$$

$$f; \bot_{YZ} = \bot_{XZ} \text{ for } f : X \to Y \tag{4.13}$$

$$\langle \bot_{XY}, \bot_{XZ}\rangle = \bot_{X,Y\times Z} : X \to Y \times Z \tag{4.14}$$

The above axiomatization consists of: the axioms (4.1)–(4.6) of categories with binary products, axioms (4.7)–(4.10) asserting that the $\leq$ is a partial order with least element $\perp$, axioms (4.11)–(4.12) stating that composition and pairing are monotone with respect to the partial order, as well as some additional axioms (4.13)–(4.14) for the interaction of $\perp$ with composition and pairing.

Example 71 (Category $\mathbf{Pposet}$)**.** A *pointed poset* is a partial order $(X, \leq)$ with a least element, which we typically denote by $\perp_X$. We call $\mathbf{Pposet}$ the category of pointed posets and monotone functions. The product $X \times Y$ of two objects X, Y is the cartesian product together with the pointwise partial order: $(x, y) \leq (x', y')$ iff $x \leq x'$ and $y \leq y'$. The least element of $X \times Y$ is the pair $\perp_{X \times Y} = (\perp_X, \perp_Y)$ of the least elements $\perp_X$ and $\perp_Y$ of X and Y respectively. The projections and the pairing operation are defined in the expected way:

$$\pi_1(x, y) = x \qquad \pi_2(x, y) = y \qquad \langle f, g\rangle(x) = (f(x), g(x)) \text{ for all } x \in X$$

for maps $f \in \mathbf{Pposet}(X, Y)$ and $g \in \mathbf{Pposet}(X, Z)$. The partial order $\leq$ on $\mathbf{Pposet}(X, Y)$ is defined pointwise: $f \leq g$ iff $f(x) \leq g(x)$ for all $x \in X$. The least element $\perp_{XY}$ of $\mathbf{Pposet}(X, Y)$ is the constant mapping given by $\perp_{XY}(x) = \perp_Y$ for all $x \in X$. We observe that all constants id_X, π_1^{XY}, π_2^{XY}, $\perp_{XY}$ are monotone maps, and the operations ; and $\langle \cdot, \cdot \rangle$ preserve monotonicity. It is easy to verify that $\mathbf{Pposet}$ is a pointed ordered category with binary products, that is, it satisfies all axioms of Definition 70.

Example 72 (Category $\mathbf{Partial}$)**.** Let A be a *base type*, and define the set of *types* $X, Y, \ldots$ to be given by the grammar $X, Y ::= A \mid X \times Y$. Let S be a nonempty *base set*, and define the *interpretation* $[\![\cdot]\!]$ of types as posets in the following way:

$$[\![A]\!] \triangleq S \cup \{\perp_S\} \qquad\qquad [\![X \times Y]\!] \triangleq [\![X]\!] \times [\![Y]\!]$$

where $\perp_S$ is a fresh symbol denoting undefinedness. The order on $[\![A]\!]$ is given by $\perp_S \leq x$ for all $x \in S$, and the order on $[\![X \times Y]\!]$ is defined componentwise from the orders on $[\![X]\!]$ and $[\![Y]\!]$. We will define now the typed structure $\mathbf{Partial}_S$ of *partial functions with lazy products* (over the base set S) . The objects of $\mathbf{Partial}_S$ are the (syntactic) types we defined before. The homset $\mathbf{Partial}_S(X, Y)$ consists of formal triples of the form $f : X \rightharpoonup Y$, where X, Y are types and f is a monotone function of type $f : [\![X]\!] \to [\![Y]\!]$. The constant elements $\mathsf{id}, \pi_1, \pi_2, \bot$ and the operations ; and $\langle \cdot, \cdot \rangle$ are defined as in $\mathbf{Pposet}$. Since $\mathbf{Partial}_S$ is (isomorphic to) a substructure of $\mathbf{Pposet}$, it is also a pointed ordered category with products.

Definition 73 (Closure Operator)**.** Let cl be a function of type $\wp X \to \wp X$, where $\wp X$ is the powerset of the set X. We say that cl is a *closure operator* on X if it is:

1. *Increasing*: $A \subseteq \mathsf{cl}(A)$ for every subset $A \subseteq X$.
2. *Monotone*: For all subsets $A, B \subseteq X$, if $A \subseteq B$ then $\mathsf{cl}(A) \subseteq \mathsf{cl}(B)$.
3. *Idempotent*: $\mathsf{cl}(\mathsf{cl}(A)) = \mathsf{cl}(A)$ for every subset $A \subseteq X$.

For an element $x \in X$, we write $\mathsf{cl}(x)$ to mean $\mathsf{cl}(\{x\})$. A subset $A \subseteq X$ is said to be *closed* (with respect to the closure operator cl) if it satisfies $\mathsf{cl}(A) = A$. Assuming that X is partially ordered by $\leq$, we say that cl *respects the order* if $x \leq y$ implies $\mathsf{cl}(x) \subseteq \mathsf{cl}(y)$ for all $x, y \in X$.

Observation 74. Let $(X, \leq)$ be a partial order and cl be a closure operator on X that respects the order. We observe that every $\mathsf{cl}(A)$ for $A \subseteq X$ is closed downwards, that is, $y \in \mathsf{cl}(A)$ and $x \leq y$ imply that $x \in \mathsf{cl}(A)$.

Let $(X, \leq)$ be a partial order and A be a subset of X. We define

$$\downarrow\! A \triangleq \{x' \in X \mid x' \leq x \text{ for some } x \in A\}.$$

If x is an element of X, we put $\downarrow\! x = \downarrow\!\{x\}$.

Example 75 (Lowersets)**.** Let X be a pointed poset. Define the operator $\mathsf{cl}_X^{\leq}$:

$\wp X \to \wp X$ as

$$\mathsf{cl}^{\leq}_X(A) \triangleq \{\bot_X\} \cup {\downarrow} A \tag{4.15}$$

for a subset $A \subseteq X$. Equivalently, we can define $\mathsf{cl}^{\leq}_X(A)$ to be the smallest nonempty subset of X that contains A and is closed downwards. It is easy to see that $\mathsf{cl}^{\leq}_X$ is a closure operator that respects the order $\leq$ of X. The closed sets w.r.t. to $\mathsf{cl}^{\leq}_X$ are the nonempty subsets of X that are closed downwards. For an element $x \in X$ in particular, we have that $\mathsf{cl}^{\leq}_X(x) = {\downarrow} x$.

As mentioned before, we want to model nondeterminism using "closed sets" that describe the available nondeterministic choices. Below, we put forward a list of desired properties for closure operators cl on the homsets of a base category $\mathcal{C}$. These properties say that the operators cl give rise to a reasonable notion of nondeterminism.

Definition 76 (Category With Closure Operator)**.** Let $\mathcal{C}$ be a pointed ordered category with products, and suppose that we are given a closure operator cl_{XY} on every homset $\mathcal{C}(X,Y)$. We say that the operators cl *interact well* with the category $\mathcal{C}$ if the following are satisfied:

1. Every operator cl_{XY} respects the order of $\mathcal{C}(X,Y)$ (see Definition 73).
2. For all elements $f, g \in \mathcal{C}(X,Y)$, $\mathsf{cl}_{XY}(f) = \mathsf{cl}_{XY}(g)$ implies that $f = g$.
3. The closure operators *interact well* with the composition of the category:
$$\frac{\emptyset \neq F \subseteq \mathcal{C}(X,Y) \qquad \emptyset \neq G \subseteq \mathcal{C}(Y,Z)}{f \in \mathsf{cl}(F) \wedge g \in \mathsf{cl}(G) \implies f;g \in \mathsf{cl}(F;G)}.$$
4. The closure operators *interact well* with the binary products of the category:
$$\frac{\emptyset \neq F \subseteq \mathcal{C}(X,Y) \qquad \emptyset \neq G \subseteq \mathcal{C}(X,Z)}{f \in \mathsf{cl}(F) \wedge g \in \mathsf{cl}(G) \implies \langle f,g\rangle \in \mathsf{cl}(\langle F,G\rangle)}.$$

Notation. We have used above composition and pairing operations lifted to sets of morphisms:

$$\frac{F \subseteq \mathcal{C}(X,Y) \qquad G \subseteq \mathcal{C}(Y,Z)}{F;G \triangleq \{f;g \mid f \in F \text{ and } g \in G\}} \qquad \frac{F \subseteq \mathcal{C}(X,Y) \qquad G \subseteq \mathcal{C}(X,Z)}{\langle F,G\rangle \triangleq \{\langle f,g\rangle \mid f \in F \text{ and } g \in G\}}$$

Moreover, we use the abbreviations $f;G = \{f\};G$ and $F;g = F;\{g\}$.

Let us try to give now an intuitive explanation for some of the properties listed in the above definition. We demand that cl respects the order, because we want the nondeterministic choice operation to be compatible with the "less defined than" order $\leq$ of the category $\mathcal{C}$. Property 3 about the interaction between cl and composition ; says informally the following: Suppose that one possible way to resolve the nondeterminism of the nondeterministic programs ϕ and ψ is described by the (deterministic) $\mathcal{C}$-arrows f and g. Then, the $\mathcal{C}$-arrow $f;g$ is a possible way to resolve the nondeterminism of the composite nondeterministic program ϕ; ψ.

Lemma 77 (Lowersets Interact Well)**.** Let $\mathcal{C}$ be a pointed ordered category with products. The family $\mathsf{cl}^{\leq}$ of lowerset closure operators $\mathsf{cl}^{\leq}_{XY}$ (see Example 75 for the definition) on every homset $\mathcal{C}(X,Y)$ interacts well with $\mathcal{C}$.

Proof. The homset $\mathcal{C}(X,Y)$ is partially ordered by $\leq$ and its least element is $\perp_{XY}$. First, let us recall the definition of the operator $\mathsf{cl}^{\leq}_{XY}$ on $\mathcal{C}(X,Y)$:

$$\mathsf{cl}^{\leq}_{XY}(F) = \{\perp_{XY}\} \cup {\downarrow}F = \{\perp_{XY}\} \cup \{f' \in \mathcal{C}(X,Y) \mid f' \leq f \text{ for some } f \in F\}$$

for a subset $F \subseteq \mathcal{C}(X,Y)$. For the particular case where F is nonempty, the definition becomes $\mathsf{cl}^{\leq}_{XY}(F) = {\downarrow}F$. We have already discussed in Example 75 that $\mathsf{cl}^{\leq}_{XY}$ is a closure operator that respects the order. For all elements f and g of $\mathcal{C}(X,Y)$, the equality $\mathsf{cl}(f) = \mathsf{cl}(g)$ means that ${\downarrow}f = {\downarrow}g$, which in turn implies that $f = g$.

Now, we show that $\mathsf{cl}^{\leq}$ interacts well with composition. Consider nonempty subsets $F \subseteq \mathcal{C}(X, Y)$ and $G \subseteq \mathcal{C}(Y, Z)$ and elements $f' \in \mathsf{cl}^{\leq}(F)$ and $g' \in \mathsf{cl}^{\leq}(G)$. There are $f \in F$ and $g \in G$ such that $f' \leq f$ and $g' \leq g$. It follows that $f; g \in F; G \subseteq \mathsf{cl}^{\leq}(F; G)$ and $f'; g' \leq f; g$ by monotonicity of composition. Since $\mathsf{cl}^{\leq}(F; G)$ is closed downwards, we conclude that $f'; g'$ is also in $\mathsf{cl}^{\leq}(F; G)$. We can show analogously that $\mathsf{cl}^{\leq}$ interacts well with the pairing operation $\langle \cdot, \cdot \rangle$ making use of the fact that pairing is monotone in both arguments. □

As we outlined before, a base category $\mathcal{C}$ and an appropriate family cl of closure operators on the homsets of $\mathcal{C}$ are sufficient to model a kind of nondeterminism. We give now the precise definition of the category $\mathcal{C}_{\mathsf{cl}}$ of nondeterministic arrows that corresponds to our previous intuitive descriptions.

Definition 78 (Nondeterministic Structure from Closure Operators)**.** Let $\mathcal{C}$ be a pointed ordered category with products, and cl_{XY} be a family of closure operations that interact well with $\mathcal{C}$. We define the typed structure $\mathcal{C}_{\mathsf{cl}}$ where each homset $\mathcal{C}_{\mathsf{cl}}(X, Y)$ is the set of all closed subsets of $\mathcal{C}(X, Y)$, together with the following constants and operations:

$$\begin{array}{rl}
\eta_X \triangleq \mathsf{cl}(\mathsf{id}_X) & F \mathbin{;} G \triangleq \mathsf{cl}(F; G) \text{ for } F \subseteq \mathcal{C}(X, Y) \text{ and } G \subseteq \mathcal{C}(Y, Z) \\
⫫_{XY} \triangleq \mathsf{cl}(\perp_{XY}) & \langle\!\langle F, G \rangle\!\rangle \triangleq \mathsf{cl}(\langle F, G \rangle) \text{ for } F \subseteq \mathcal{C}(X, Y) \text{ and } G \subseteq \mathcal{C}(X, Z) \\
\varpi_1^{XY} \triangleq \mathsf{cl}(\pi_1^{XY}) & \sum_{i \in I} F_i \triangleq \mathsf{cl}(\bigcup_{i \in I} F_i) \text{ for } F_i \subseteq \mathcal{C}(X, Y) \text{ with } i \in I \\
\varpi_2^{XY} \triangleq \mathsf{cl}(\pi_2^{XY}) &
\end{array}$$

Every homset $\mathcal{C}_{\mathsf{cl}}$ is partially ordered by $\subseteq$. Finally, define D_{XY} to be the subset of $\mathcal{C}_{\mathsf{cl}}(X, Y)$ that contains the sets of the form $\mathsf{cl}(f)$ where $f \in \mathcal{C}(X, Y)$. We think of D_{XY} as a unary predicate consisting of the *deterministic arrows* of type $X \rightsquigarrow Y$.

The axiomatic definition of a “nondeterministic category with products and

product	$(X, Y) \mapsto X \times Y$	left projection	$\varpi_1^{XY} : X \times Y \rightsquigarrow X$
identity	$\eta_X : X \rightsquigarrow X$	right projection	$\varpi_2^{XY} : X \times Y \rightsquigarrow Y$
bottom	$\perp\!\!\!\perp_{XY} : X \rightsquigarrow Y$		

composition $$\frac{\phi : X \rightsquigarrow Y \qquad \psi : Y \rightsquigarrow Z}{\phi; \psi : X \rightsquigarrow Z}$$

pairing $$\frac{\phi : X \rightsquigarrow Y \qquad \psi : X \rightsquigarrow Z}{\langle\!\langle \phi, \psi \rangle\!\rangle : X \rightsquigarrow Y \times Z}$$

product $$\frac{\phi : X \rightsquigarrow Y \qquad \psi : X' \rightsquigarrow Y'}{\phi \otimes \psi \triangleq \langle\!\langle \varpi_1; \phi, \varpi_2; \psi \rangle\!\rangle : X \times X' \rightsquigarrow Y \times Y'}$$

arbitrary sum $$\frac{\textit{nonempty} \text{ collection } \Phi \text{ of morphisms } X \rightsquigarrow Y}{\sum \Phi : X \rightsquigarrow Y}$$

indexed sum $$\frac{\phi_i : X \rightsquigarrow Y \qquad \text{index set } I}{\sum_{i \in I} \phi_i \triangleq \sum\{\phi_i \mid i \in I\} : X \rightsquigarrow Y}$$

binary sum $$\frac{\phi, \psi : X \rightsquigarrow Y}{\phi + \psi \triangleq \sum\{\phi, \psi\} : X \rightsquigarrow Y}$$

iteration $$\frac{\phi : X \rightsquigarrow X}{\phi^* \triangleq \sum_{n \geq 0} \phi^n} \quad \text{where } \phi^0 \triangleq \eta_X \text{ and } \phi^{n+1} \triangleq \phi^n; \phi$$

Figure 4.2: Constants and operations for nondeterministic categories with products and joins.

joins" that we give below (Definition 79) collects the main properties that we expect to be satisfied by nondeterministic programs with non-strict pairs. We have chosen only those properties that are relevant for our later development. In particular, the given axiomatization is sufficiently strong to see that the *deterministic subcategory* of a nondeterministic category is a model of the axiomatization of Definition 70.

Definition 79 (Nondeterministic Category)**.** Consider a typed algebraic structure with the operations of Figure 4.2 and a unary predicate D on every homset. We say that the structure is a *nondeterministic category with (binary) products and (arbitrary) joins* if it is a model of the following infinitary universal Horn axioms:

$$D(\eta_X) \qquad D(\varpi_1^{XY}) \qquad D(\varpi_2^{XY}) \qquad D(\perp\!\!\!\perp_{XY}) \tag{4.16}$$

$$D(\phi) \wedge D(\psi) \implies D(\phi;\psi) \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi : Y \rightsquigarrow Z \tag{4.17}$$

$$D(\phi) \wedge D(\psi) \implies D(\langle\!\langle \phi, \psi \rangle\!\rangle) \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi : X \rightsquigarrow Z \tag{4.18}$$

$$(\phi;\psi);\chi = \phi;(\psi;\chi) \text{ for } \phi : X \rightsquigarrow Y,\ \psi : Y \rightsquigarrow Z \text{ and } \chi : Z \rightsquigarrow W \tag{4.19}$$

$$\eta_X;\phi = \phi \text{ for } \phi : X \rightsquigarrow Y \tag{4.20}$$

$$\phi;\eta_Y = \phi \text{ for } \phi : X \rightsquigarrow Y \tag{4.21}$$

$$\langle\!\langle \phi, \psi \rangle\!\rangle;\varpi_1 = \phi \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi : X \rightsquigarrow Z \tag{4.22}$$

$$\langle\!\langle \phi, \psi \rangle\!\rangle;\varpi_2 = \psi \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi : X \rightsquigarrow Z \tag{4.23}$$

$$\langle\!\langle \chi;\varpi_1, \chi;\varpi_2 \rangle\!\rangle \geq \chi \text{ for } \chi : X \rightsquigarrow Y \times Z \tag{4.24}$$

$$D(\chi) \implies \langle\!\langle \chi;\varpi_1, \chi;\varpi_2 \rangle\!\rangle = \chi \text{ for } \chi : X \rightsquigarrow Y \times Z \tag{4.25}$$

$$D(\phi) \implies \phi;\langle\!\langle \psi_1, \psi_2 \rangle\!\rangle = \langle\!\langle \phi;\psi_1, \phi;\psi_2 \rangle\!\rangle \tag{4.26}$$
$$\text{for } \phi : X \rightsquigarrow Y \text{ and } \psi_i : Y \rightsquigarrow Z_i$$

$$\langle\!\langle \phi_1, \phi_2 \rangle\!\rangle;(\psi_1 \otimes \psi_2) = \langle\!\langle \phi_1;\psi_1, \phi_2;\psi_2 \rangle\!\rangle \text{ for } \phi_i : X \rightsquigarrow Y_i \text{ and } \psi_i : Y_i \rightsquigarrow Z_i \tag{4.27}$$

$$\langle\!\langle \phi, \psi \rangle\!\rangle;\langle\!\langle \varpi_2, \varpi_1 \rangle\!\rangle = \langle\!\langle \psi, \phi \rangle\!\rangle \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi : X \rightsquigarrow Z \tag{4.28}$$

$$\langle\!\langle \phi, \varpi_2 \rangle\!\rangle;\langle\!\langle \psi, \varpi_2 \rangle\!\rangle = \langle\!\langle \langle\!\langle \phi, \varpi_2 \rangle\!\rangle;\psi, \varpi_2 \rangle\!\rangle \text{ for } \phi, \psi : X \times Y \rightsquigarrow X \tag{4.29}$$

$$\perp\!\!\!\perp_{XY} \leq \phi \text{ for } \phi : X \rightsquigarrow Y \tag{4.30}$$

$$\phi;\perp\!\!\!\perp_{YZ} = \perp\!\!\!\perp_{XZ} \text{ for } \phi : X \rightsquigarrow Y \tag{4.31}$$

$$\langle\!\langle \perp\!\!\!\perp_{XY}, \perp\!\!\!\perp_{XZ} \rangle\!\rangle = \perp\!\!\!\perp_{X,Y\times Z} \tag{4.32}$$

$$\textstyle\sum\{\phi\} = \phi \text{ for } \phi : X \rightsquigarrow Y \tag{4.33}$$

$$\textstyle\sum_i(\sum_j \phi_{ij}) = \sum_{i,j} \phi_{ij} \text{ for } \phi_{ij} : X \rightsquigarrow Y \tag{4.34}$$

$$\textstyle(\sum_i \phi_i);\psi = \sum_i \phi_i;\psi \text{ for } \phi_i : X \rightsquigarrow Y \text{ and } \psi : Y \rightsquigarrow Z \tag{4.35}$$

$$\textstyle\phi;(\sum_i \psi_i) = \sum_i \phi;\psi_i \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi_i : Y \rightsquigarrow Z \tag{4.36}$$

$$\textstyle\langle\!\langle \sum_i \phi_i, \psi \rangle\!\rangle = \sum_i \langle\!\langle \phi_i, \psi \rangle\!\rangle \text{ for } \phi_i : X \rightsquigarrow Y \text{ and } \psi : X \rightsquigarrow Z \tag{4.37}$$

$$\textstyle\langle\!\langle \phi, \sum_i \psi_i \rangle\!\rangle = \sum_i \langle\!\langle \phi, \psi_i \rangle\!\rangle \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi_i : X \rightsquigarrow Z \tag{4.38}$$

where the partial order $\leq$ is induced by $+$ as follows: $\phi \leq \psi$ iff $\phi + \psi = \psi$. In

Axioms (4.34)–(4.38) the indexes range over *nonempty* index sets. We say that D is the predicate of *deterministic* morphisms. Axioms (4.16) say that the constants η_X, ϖ_1^{XY}, ϖ_2^{XY} and $\perp\!\!\!\perp_{XY}$ are deterministic, and Axioms (4.17) and (4.18) say that composition and pairing preserve determinism. We stress that the operations $\times, \varpi_1, \varpi_2, \langle\!\langle \cdot, \cdot \rangle\!\rangle$ do not give rise to categorical products.

Claim 80. Every nondeterministic category with products and joins satisfies:

$$\langle\!\langle \varpi_1, \phi \rangle\!\rangle ; \langle\!\langle \varpi_1, \psi \rangle\!\rangle = \langle\!\langle \varpi_1, \langle\!\langle \varpi_1, \phi \rangle\!\rangle ; \psi \rangle\!\rangle \text{ for } \phi, \psi : X \times Y \rightsquigarrow Y \tag{4.39}$$

$$\langle\!\langle \phi, \eta_Y \rangle\!\rangle ; \langle\!\langle \psi, \varpi_2 \rangle\!\rangle = \langle\!\langle \langle\!\langle \phi, \eta_Y \rangle\!\rangle ; \psi, \eta_Y \rangle\!\rangle \text{ for } \phi : Y \rightsquigarrow X \text{ and } \psi : X \times Y \rightsquigarrow X \tag{4.40}$$

$$\langle\!\langle \eta_X, \phi \rangle\!\rangle ; \langle\!\langle \varpi_1, \psi \rangle\!\rangle = \langle\!\langle \eta_X, \langle\!\langle \eta_X, \phi \rangle\!\rangle ; \psi \rangle\!\rangle \text{ for } \phi : X \rightsquigarrow Y \text{ and } \psi : X \times Y \rightsquigarrow Y \tag{4.41}$$

$$\langle\!\langle \phi, \varpi_2 \rangle\!\rangle^* = \langle\!\langle \langle\!\langle \phi, \varpi_2 \rangle\!\rangle^* ; \varpi_1, \varpi_2 \rangle\!\rangle \text{ for all } \phi : X \times Y \rightsquigarrow X \tag{4.42}$$

Proof. We show the claim by making crucial use of the axioms (4.28) and (4.29) of nondeterministic categories. □

Theorem 81 (First Embedding Theorem)**.** Let $\mathcal{C}$ be a pointed ordered category with products, and cl_{XY} be a family of closure operators that interact well with $\mathcal{C}$. Then, the typed structure $\mathcal{C}_{\mathsf{cl}}$ is a nondeterministic category with products and joins. Moreover, the map

$$\mathsf{cl} : f \in \mathcal{C}(X, Y) \mapsto \mathsf{cl}_{XY}(f) \in \mathcal{C}_{\mathsf{cl}}(XY)$$

from the category $\mathcal{C}$ to the category $\mathcal{C}_{\mathsf{cl}}$ is an injective order-preserving homomorphism. In other words, $\mathcal{C}$ is isomorphic to the deterministic subcategory of $\mathcal{C}_{\mathsf{cl}}$.

Proof. Before showing the soundness of the axioms of Definition 79, we will establish the following useful properties:

$$\mathsf{cl}(F)\,\mathbf{;}\,\mathsf{cl}(G) = \mathsf{cl}(F;G) \text{ for } \emptyset \neq F \subseteq \mathcal{C}(X,Y) \text{ and } \emptyset \neq G \subseteq \mathcal{C}(Y,Z) \tag{4.43}$$

$$\langle\!\langle \mathsf{cl}(F), \mathsf{cl}(G) \rangle\!\rangle = \mathsf{cl}(\langle F, G\rangle) \text{ for } \emptyset \neq F \subseteq \mathcal{C}(X,Y) \text{ and } \emptyset \neq G \subseteq \mathcal{C}(X,Z) \tag{4.44}$$

$$\mathsf{cl}(\textstyle\bigcup_{i\in I}\mathsf{cl}(F_i)) = \mathsf{cl}(\bigcup_{i\in I}F_i) \text{ for } F_i \subseteq \mathcal{C}(X,Y) \text{ with } i \in I \tag{4.45}$$

For the first equation, consider nonempty subsets $F \subseteq \mathcal{C}(X,Y)$ and $G \subseteq \mathcal{C}(Y,Z)$. We expand the definition of $\mathbf{;}$ and the equation becomes $\mathsf{cl}(\mathsf{cl}(F);\mathsf{cl}(G)) = \mathsf{cl}(F;G)$. The operator cl is increasing, so we know that $F \subseteq \mathsf{cl}(F)$ and $G \subseteq \mathsf{cl}(G)$. It follows that $F;G \subseteq \mathsf{cl}(F);\mathsf{cl}(G)$ and therefore $\mathsf{cl}(F;G) \subseteq \mathsf{cl}(\mathsf{cl}(F);\mathsf{cl}(G))$ (since cl is monotone). For the reverse containment, it suffices to show that $\mathsf{cl}(F);\mathsf{cl}(G) \subseteq \mathsf{cl}(F;G)$ (because of idempotence of cl), which is a reformulation of the hypothesis that cl interacts well with the composition ; of $\mathcal{C}$. Similarly, we can also show:

$$\mathsf{cl}(F;\mathsf{cl}(G)) = \mathsf{cl}(F;G) \qquad\qquad \mathsf{cl}(\mathsf{cl}(F);G) = \mathsf{cl}(F;G)$$

With analogous arguments (involving that cl interacts well with $\langle\cdot,\cdot,\rangle$) we get:

$$\mathsf{cl}(\langle F, \mathsf{cl}(G)\rangle) = \mathsf{cl}(\langle F, G\rangle)$$
$$\mathsf{cl}(\langle \mathsf{cl}(F), G\rangle) = \mathsf{cl}(\langle F, G\rangle)$$
$$\mathsf{cl}(\langle \mathsf{cl}(F), \mathsf{cl}(G)\rangle) = \mathsf{cl}(\langle F, G\rangle)$$

for all nonempty $F \subseteq \mathcal{C}(X,Y)$ and $G \subseteq \mathcal{C}(Y,Z)$. The equation $\langle\!\langle \mathsf{cl}(F), \mathsf{cl}(G)\rangle\!\rangle = \mathsf{cl}(\langle F, G\rangle)$ follows. The third equation follows just from properties of the closure operation.

Immediately from their definitions, we see that the constants η_X, $\perp\!\!\!\perp_{XY}$, ϖ_1^{XY} and ϖ_2^{XY} are deterministic. Now, we see that the operation $\mathbf{;}$ preserves determinism: $\mathsf{cl}(f)\,\mathbf{;}\,\mathsf{cl}(g) = \mathsf{cl}(f;g)$ for $f \in \mathcal{C}(X,Y)$ and $g \in \mathcal{C}(Y,Z)$. Similarly, we can show that the pairing operation $\langle\cdot,\cdot\rangle$ preserves determinism: $\langle\!\langle \mathsf{cl}(f), \mathsf{cl}(g)\rangle\!\rangle = \mathsf{cl}(\langle f, g\rangle)$ for $f \in \mathcal{C}(X,Y)$ and $g \in \mathcal{C}(X,Z)$.

Using the properties that we proved in the first paragraph of the proof, we continue to show the soundness of axioms (4.19)–(4.38). We expand the definition of the partial order for elements ϕ and ψ of $\mathcal{C}_{\mathsf{cl}}(X,Y)$: $\phi \leq \psi$ iff $\phi + \psi = \psi$ iff $\sum\{\phi,\psi\} = \psi$ iff $\mathsf{cl}(\phi \cup \psi) = \psi$ iff $\phi \cup \psi \subseteq \psi$ iff $\phi \subseteq \psi$. We write F, F_i, G, G_i, H below to range over nonempty subsets of $\mathcal{C}$-morphisms of the appropriate type, and f, h to range over $\mathcal{C}$-morphisms.

$$\begin{aligned}
(F\mathbf{;}G)\mathbf{;}H &= \mathsf{cl}(F;G)\mathbf{;}H = \mathsf{cl}(\mathsf{cl}(F;G);H)\\
&= \mathsf{cl}((F;G);H) = \mathsf{cl}(F;(G;H))\\
F\mathbf{;}(G\mathbf{;}H) &= F\mathbf{;}\mathsf{cl}(G;H) = \mathsf{cl}(F;\mathsf{cl}(G;H)) = \mathsf{cl}(F;(G;H))\\
\eta_X\mathbf{;}F &= \mathsf{cl}(\mathsf{id}_X)\mathbf{;}F = \mathsf{cl}(\mathsf{cl}(\mathsf{id}_X);F) = \mathsf{cl}(\mathsf{id}_X;F) = \mathsf{cl}(F)\\
F\mathbf{;}\eta_Y &= F\mathbf{;}\mathsf{cl}(\mathsf{id}_Y) = \mathsf{cl}(F;\mathsf{cl}(\mathsf{id}_Y)) = \mathsf{cl}(F;\mathsf{id}_Y) = \mathsf{cl}(F)\\
\langle\!\langle F,G\rangle\!\rangle\mathbf{;}\varpi_1 &= \mathsf{cl}(\langle F,G\rangle)\mathbf{;}\mathsf{cl}(\pi_1) = \mathsf{cl}(\langle F,G\rangle;\pi_1) = \mathsf{cl}(F)\\
\langle\!\langle H\mathbf{;}\varpi_1, H\mathbf{;}\varpi_2\rangle\!\rangle &= \langle\!\langle H\mathbf{;}\mathsf{cl}(\pi_1), H\mathbf{;}\mathsf{cl}(\pi_2)\rangle\!\rangle = \langle\!\langle \mathsf{cl}(H;\mathsf{cl}(\pi_1)), \mathsf{cl}(H;\mathsf{cl}(\pi_2))\rangle\!\rangle\\
&= \langle\!\langle \mathsf{cl}(H;\pi_1), \mathsf{cl}(H;\pi_2)\rangle\!\rangle = \mathsf{cl}(\langle H;\pi_1, H;\pi_2\rangle) \supseteq \mathsf{cl}(H)\\
\langle\!\langle \mathsf{cl}(h)\mathbf{;}\varpi_1, \mathsf{cl}(h)\mathbf{;}\varpi_2\rangle\!\rangle &= \langle\!\langle \mathsf{cl}(h)\mathbf{;}\mathsf{cl}(\pi_1), \mathsf{cl}(h)\mathbf{;}\mathsf{cl}(\pi_2)\rangle\!\rangle = \langle\!\langle \mathsf{cl}(h;\pi_1), \mathsf{cl}(h;\pi_2)\rangle\!\rangle\\
&= \mathsf{cl}(\langle h;\pi_1, h;\pi_2\rangle) = \mathsf{cl}(h)\\
\mathsf{cl}(f)\mathbf{;}\langle\!\langle G_1,G_2\rangle\!\rangle &= \mathsf{cl}(f)\mathbf{;}\mathsf{cl}(\langle G_1,G_2\rangle) = \mathsf{cl}(f;\langle G_1,G_2\rangle) = \mathsf{cl}(\langle f;G_1, f;G_2\rangle)\\
\langle\!\langle \mathsf{cl}(f)\mathbf{;}G_1, \mathsf{cl}(f)\mathbf{;}G_2\rangle\!\rangle &= \langle\!\langle \mathsf{cl}(\mathsf{cl}(f);G_1), \mathsf{cl}(\mathsf{cl}(f);G_2)\rangle\!\rangle = \langle\!\langle \mathsf{cl}(f;G_1), \mathsf{cl}(f;G_2)\rangle\!\rangle\\
&= \mathsf{cl}(\langle f;G_1, f;G_2\rangle)\\
\langle\!\langle F_1,F_2\rangle\!\rangle\mathbf{;}(G_1\otimes G_2) &= \langle\!\langle F_1,F_2\rangle\!\rangle\mathbf{;}\langle\!\langle \varpi_1\mathbf{;}G_1, \varpi_2\mathbf{;}G_2\rangle\!\rangle\\
&= \mathsf{cl}(\langle F_1,F_2\rangle)\mathbf{;}\langle\!\langle \mathsf{cl}(\pi_1;G_1), \mathsf{cl}(\pi_2;G_2)\rangle\!\rangle\\
&= \mathsf{cl}(\langle F_1,F_2\rangle)\mathbf{;}\mathsf{cl}(\langle \pi_1;G_1, \pi_2;G_2\rangle)\\
&= \mathsf{cl}(\langle F_1,F_2\rangle;\langle \pi_1;G_1, \pi_2;G_2\rangle)\\
&= \mathsf{cl}(\langle F_1;G_1, F_2;G_2\rangle)
\end{aligned}$$

$$
\begin{aligned}
\langle\!\langle F_1 \boldsymbol{;} G_1, F_2 \boldsymbol{;} G_2 \rangle\!\rangle &= \langle\!\langle \mathsf{cl}(F_1; G_1), \mathsf{cl}(F_2; G_2) \rangle\!\rangle = \mathsf{cl}(\langle F_1; G_1, F_2; G_2 \rangle) \\
\langle\!\langle F, G \rangle\!\rangle \boldsymbol{;} \langle\!\langle \varpi_2, \varpi_1 \rangle\!\rangle &= \mathsf{cl}(\langle F, G \rangle) \boldsymbol{;} \mathsf{cl}(\langle \pi_2, \pi_1 \rangle) = \mathsf{cl}(\langle F, G \rangle; \langle \pi_2, \pi_1 \rangle) \\
&= \mathsf{cl}(\langle G, F \rangle) = \langle\!\langle G, F \rangle\!\rangle \\
\langle\!\langle F, \varpi_2 \rangle\!\rangle \boldsymbol{;} \langle\!\langle G, \varpi_2 \rangle\!\rangle &= \mathsf{cl}(\langle F, \pi_2 \rangle) \boldsymbol{;} \mathsf{cl}(\langle G, \pi_2 \rangle) = \mathsf{cl}(\langle F, \pi_2 \rangle; \langle G, \pi_2 \rangle) \\
&= \mathsf{cl}(\langle \langle F, \pi_2 \rangle; G, \pi_2 \rangle) \\
\langle\!\langle \langle\!\langle F, \varpi_2 \rangle\!\rangle \boldsymbol{;} G, \varpi_2 \rangle\!\rangle &= \langle\!\langle \mathsf{cl}(\langle F, \pi_2 \rangle) \boldsymbol{;} G, \varpi_2 \rangle\!\rangle = \langle\!\langle \mathsf{cl}(\langle F, \pi_2 \rangle; G), \varpi_2 \rangle\!\rangle \\
&= \mathsf{cl}(\langle \langle F, \pi_2 \rangle; G, \pi_2 \rangle) \\
\mathsf{cl}(F) &= \mathsf{cl}({\textstyle\bigcup_{f \in F}} \{f\}) = \mathsf{cl}({\textstyle\bigcup_{f \in F}} \mathsf{cl}(f)) \\
&\supseteq \mathsf{cl}({\textstyle\bigcup_{f \in F}} \mathsf{cl}(\perp_{XY})) = \mathsf{cl}(\perp_{XY}) \\
F \boldsymbol{;} \perp\!\!\!\perp_{YZ} &= F \boldsymbol{;} \mathsf{cl}(\perp_{YZ}) = \mathsf{cl}(F; \mathsf{cl}(\perp_{YZ})) = \mathsf{cl}(F; \perp_{YZ}) \\
&= \mathsf{cl}(\perp_{XZ}) = \perp\!\!\!\perp_{XZ} \\
\langle\!\langle \perp\!\!\!\perp_{XY}, \perp\!\!\!\perp_{XZ} \rangle\!\rangle &= \langle\!\langle \mathsf{cl}(\perp_{XY}), \mathsf{cl}(\perp_{XZ}) \rangle\!\rangle = \mathsf{cl}(\langle \perp_{XY}, \perp_{XZ} \rangle) \\
&= \mathsf{cl}(\perp_{X, Y \times Z}) = \perp\!\!\!\perp_{X, Y \times Z} \\
{\textstyle\sum} \{\mathsf{cl}(F)\} &= \mathsf{cl}(\mathsf{cl}(F)) = \mathsf{cl}(F) \\
{\textstyle\sum_i} ({\textstyle\sum_j} \mathsf{cl}(F_{ij})) &= {\textstyle\sum_i} \mathsf{cl}({\textstyle\bigcup_j} \mathsf{cl}(F_{ij})) = {\textstyle\sum_i} \mathsf{cl}({\textstyle\bigcup_j} F_{ij}) = \mathsf{cl}({\textstyle\bigcup_i} \mathsf{cl}({\textstyle\bigcup_j} F_{ij})) \\
&= \mathsf{cl}({\textstyle\bigcup_i \bigcup_j} F_{ij}) = \mathsf{cl}({\textstyle\bigcup_{i,j}} F_{ij}) \\
{\textstyle\sum_{i,j}} \mathsf{cl}(F_{ij}) &= \mathsf{cl}({\textstyle\bigcup_{i,j}} \mathsf{cl}(F_{ij})) = \mathsf{cl}({\textstyle\bigcup_{i,j}} F_{ij}) \\
({\textstyle\sum_i} F_i) \boldsymbol{;} G &= \mathsf{cl}({\textstyle\bigcup_i} F_i) \boldsymbol{;} G = \mathsf{cl}(\mathsf{cl}({\textstyle\bigcup_i} F_i); G) = \mathsf{cl}(({\textstyle\bigcup_i} F_i); G) = \mathsf{cl}({\textstyle\bigcup_i} F_i; G) \\
{\textstyle\sum_i} F_i \boldsymbol{;} G &= {\textstyle\sum_i} \mathsf{cl}(F_i; G) = \mathsf{cl}({\textstyle\bigcup_i} \mathsf{cl}(F_i; G)) = \mathsf{cl}({\textstyle\bigcup_i} F_i; G) \\
\langle\!\langle {\textstyle\sum_i} F_i, G \rangle\!\rangle &= \langle\!\langle \mathsf{cl}({\textstyle\bigcup_i} F_i), G \rangle\!\rangle = \mathsf{cl}(\langle \mathsf{cl}({\textstyle\bigcup_i} F_i), G \rangle) \\
&= \mathsf{cl}(\langle {\textstyle\bigcup_i} F_i, G \rangle) = \mathsf{cl}({\textstyle\bigcup_i} \langle F_i, G \rangle) \\
{\textstyle\sum_i} \langle\!\langle F_i, G \rangle\!\rangle &= {\textstyle\sum_i} \mathsf{cl}(\langle F_i, G \rangle) = \mathsf{cl}({\textstyle\bigcup_i} \mathsf{cl}(\langle F_i, G \rangle)) = \mathsf{cl}({\textstyle\bigcup_i} \langle F_i, G \rangle)
\end{aligned}
$$

So, $\mathcal{C}_{\mathsf{cl}}$ is a nondeterministic category with binary products and arbitrary joins.

That the map cl is injective follows immediately from Condition (2) in Definition 76. The map cl is order-preserving because it respects the order: $f \leq g$ implies that $\mathsf{cl}(f) \subseteq \mathsf{cl}(g)$. To show that cl is a homomorphism, we have to see that it preserves the constants and operations:

$$\begin{array}{lll} \mathsf{cl}(\mathsf{id}_X) = \eta_X & \mathsf{cl}(\pi_1^{XY}) = \varpi_1^{XY} & \mathsf{cl}(f;g) = \mathsf{cl}(f);\mathsf{cl}(g) \\ \mathsf{cl}(\bot_{XY}) = \perp\!\!\!\perp_{XY} & \mathsf{cl}(\pi_2^{XY}) = \varpi_2^{XY} & \mathsf{cl}(\langle f, g\rangle) = \langle\!\langle \mathsf{cl}(f), \mathsf{cl}(g) \rangle\!\rangle \end{array}$$

The above equations follow from the definition of the constants for $\mathcal{C}_{\mathsf{cl}}$ and from the properties of cl that we established in the beginning of the proof. □

Corollary 82 (Embedding With Lowersets)**.** Let $\mathcal{C}$ be a pointed ordered category with products, and $\mathsf{cl}^{\leq}$ be the family of lowerset closure operators on $\mathcal{C}$. Then, $\mathcal{C}_{\mathsf{cl}^{\leq}}$ is a nondeterministic category with products and joins, and the map $f \mapsto \mathsf{cl}(f)$ embeds $\mathcal{C}$ into $\mathcal{C}_{\mathsf{cl}^{\leq}}$.

Proof. Immediately from Theorem 81 and Lemma 77. □

The main result of this section (Theorem 81 and Corollary 82) is a model-theoretic *conservative extension* of an arbitrary pointed ordered category with products into a structure that possesses nondeterminism. In the next section we will consider the case where the base category is ω-continuous, which induces an additional *parametric least fixpoint* operation.

4.3 Continuity, Least Fixpoints and Nondeterminism

In the previous section we showed how a very general class of ordered typed structures with products can be conservatively extended with nondeterminism. Here we restrict our attention to the smaller class of *ω-continuous* typed structures, whose homsets are ω-CPOs and composition is ω-continuous in both arguments. This subclass is particularly interesting because in every such ω-continuous category we can define a *least fixpoint* operation in terms of suprema of countable chains. In particular, our standard model **CPO** (the category of ω-CPOs and ω-continuous functions) belongs to this subclass. We will describe in this section a new model-theoretic construction that conservatively extends any ω-continuous category with nondeterministic structure. We make use of some of the results of §4.2, but the setting of this section is more specific and we can thus prove more useful results.

An *ω-complete partial order* (ω-CPO) is a pointed poset $(X, \leq)$ with least element $\bot_X$ that is *ω-complete*: every ω-chain (countably infinite chain) $x_0 \leq x_1 \leq \cdots$ has a supremum $\sup_i x_i$ in X. If X, Y are ω-CPOs, then so is their cartesian product $X \times Y$ under the componentwise order. For an ω-chain $(x_0, y_0) \leq (x_1, y_1) \leq (x_2, y_2) \leq \cdots$ in $X \times Y$ we have that

$$\sup_i (x_i, y_i) = (\sup_i x_i, \sup_i y_i).$$

A function $f : X \to Y$ between ω-CPOs X and Y is called *ω-continuous* if it preserves suprema of ω-chains: for every ω-chain $x_0 \leq x_1 \leq x_2 \leq \cdots$ in X, it holds that $f(\sup_i x_i) = \sup_i f(x_i)$. It is easy to see that every ω-continuous function is monotone.

Definition 83. An *ω-continuous category with (binary) products* is a pointed ordered category $\mathcal{C}$ with binary products whose homsets are ω-CPOs and satisfies additionally the equations:

$$(\sup_i f_i); g = \sup_i (f_i; g) \text{ for } \omega\text{-chain } f_i : X \to Y \text{ and } g : Y \to Z \qquad (4.46)$$

$$f; (\sup_i g_i) = \sup_i (f; g_i) \text{ for } f : X \to Y \text{ and } \omega\text{-chain } g_i : Y \to Z \qquad (4.47)$$

The above equations say that composition is ω-continuous in both arguments.

Observation 84 (Pairing Is Continuous)**.** Let $\mathcal{C}$ be an ω-continuous category with products. We claim that the pairing operation $\langle \cdot, \cdot \rangle$ is ω-continuous in both arguments:

$$\langle \sup_i f_i, g \rangle = \sup_i \langle f_i, g \rangle \text{ for } \omega\text{-chain } f_i : X \to Y \text{ and } g : X \to Z \tag{4.48}$$

$$\langle f, \sup_i g_i \rangle = \sup_i \langle f, g_i \rangle \text{ for } f : X \to Y \text{ and } \omega\text{-chain } g_i : X \to Z \tag{4.49}$$

We show how to derive Equation 4.48 using the axioms we stipulated for $\mathcal{C}$:

$$(\sup_i \langle f_i, g \rangle); \pi_1 = \sup_i (\langle f_i, g \rangle; \pi_1) = \sup_i f_i$$

$$(\sup_i \langle f_i, g \rangle); \pi_2 = \sup_i (\langle f_i, g \rangle; \pi_2) = \sup_i g = g$$

which imply the desired equation by the uniqueness axiom (4.6) for pairing. The proof for Equation 4.49 is analogous and we therefore omit it.

Example 85 (Category **CPO**)**.** For ω-CPOs X and Y, we denote by $[X \to Y]$ the ω-CPO of all ω-continuous functions from X to Y ordered pointwise. We have to verify that $[X \to Y]$ is closed under suprema of ω-chains. The supremum of an ω-chain $f_0 \leq f_1 \leq f_2 \leq \cdots$ in $[X \to Y]$ is $\sup_i f_i = \lambda x \in X.\ \sup_i f_i(x)$ and it is ω-continuous:

$$\begin{aligned}
(\sup_i f_i)(\sup_j x_j) &= \sup_i (f_i(\sup_j x_j)) && [\text{definition of } \sup_i f_i] \\
&= \sup_i \sup_j f_i(x_j) && [\text{every } f_i \text{ is } \omega\text{-continuous}] \\
&= \sup_j \sup_i f_i(x_j) && [\text{interchange sups}] \\
&= \sup_j (\sup_i f_i)(x_j) && [\text{definition of } \sup_i f_i]
\end{aligned}$$

for a chain $x_0 \leq x_1 \leq x_2 \leq \cdots$ in X. The ω-continuous functions on ω-CPOs are closed under well-typed composition and pairing and contain all identities and projections. Moreover, composition is ω-continuous in both arguments. Thus, ω-CPOs and ω-continuous maps form an ω-continuous category with binary products denoted **CPO**.

Definition 86 (Ideals and Closure)**.** Let X be an ω-CPO. A subset $I \subseteq X$ is called an *ideal* of X if it is nonempty, closed downwards, and closed under suprema of ω-chains. We define the operator $\mathsf{cl}^{\mathfrak{I}}_X : \wp X \to \wp X$ as follows: $\mathsf{cl}^{\mathfrak{I}}_X(A)$ is the smallest ideal of X that contains A. For an element $x \in X$ we have that $\mathsf{cl}^{\mathfrak{I}}_X(x) = \downarrow x$, because $\downarrow x$ is closed under suprema of ω-chains. Immediately from the definition of $\mathsf{cl}^{\mathfrak{I}}_X$, we obtain that it is a closure operator that respects the order. For a subset $A \subseteq X$, define $\overline{\sup}(A)$ to be the set of suprema of all ω-chains in A:

$$\overline{\sup}(A) \triangleq \{\sup_i x_i \mid \omega\text{-chain } (x_i)_{i<\omega} \text{ in } A\}. \tag{4.50}$$

Now, we define the operator τ on X as $\tau(A) = \downarrow A \cup \overline{\sup}(A)$ for $A \subseteq X$ and the sequence:

$$\tau_0(A) \triangleq \{\bot_X\} \cup A \qquad \tau_{\alpha+1}(A) \triangleq \tau(\tau_\alpha(A)) \qquad \tau_\lambda(A) \triangleq \textstyle\bigcup_{\alpha<\lambda} \tau_\alpha(A) \tag{4.51}$$

for a limit ordinal λ. It holds that $A \subseteq \tau(A)$ and therefore the transfinite sequence $(\tau_\alpha(A))_{\alpha \in \mathbf{Ord}}$ is increasing: $\alpha \leq \beta$ implies $\tau_\alpha(A) \subseteq \tau_\beta(A)$. Finally, we have the equivalent definition

$$\mathsf{cl}^{\mathfrak{I}}_X(A) = \textstyle\bigcup_{\alpha \in \mathbf{Ord}} \tau_\alpha(A),$$

where **Ord** is the class of all ordinals.

Lemma 87 (Ideals Interact Well)**.** Let $\mathcal{C}$ be an ω-continuous category with products. The family $\mathsf{cl}^{\mathfrak{I}}$ of closure operators $\mathsf{cl}^{\mathfrak{I}}_{XY}$ on every homset $\mathcal{C}(X, Y)$ interacts well with $\mathcal{C}$ (Definition 76).

Proof. We introduced the definition of the ideal closure operator $\mathsf{cl}^{\mathfrak{I}}$ in Definition 86. We argue as in Lemma 77 that $\mathsf{cl}^{\mathfrak{I}}$ respects the order, and also that $\mathsf{cl}^{\mathfrak{I}}(f) = \mathsf{cl}^{\mathfrak{I}}(g)$ implies $f = g$.

Now, we have to show that $\mathsf{cl}^{\mathfrak{I}}$ interacts well with composition. Consider arbitrary nonempty subsets $F \subseteq \mathcal{C}(X, Y)$ and $G \subseteq \mathcal{C}(Y, Z)$. Since the sets F

and G are nonempty, we can consider here a slightly simpler definition for the sequence τ_α than the one of Equation (4.51), with the only difference being that now $\tau_0 = A$. We will establish the claim: for every ordinal α,

$$f \in \tau_\alpha(F) \wedge g \in \tau_\alpha(G) \implies f;g \in \tau_\alpha(F;G).$$

The proof is by transfinite induction. The base case is trivial: $f \in \tau_0(F) = F$ and $g \in \tau_0(G) = G$ imply that $f;g \in F;G = \tau_0(F;G)$. For the case of a successor ordinal, we suppose that $f \in \tau_{\alpha+1}(F) = \tau(\tau_\alpha(F))$ and $g \in \tau_{\alpha+1}(G) = \tau(\tau_\alpha(G))$, and we examine cases:

1. Case: $f \leq f'$ for some $f' \in \tau_\alpha(F)$ and $g \leq g'$ for some $g' \in \tau_\alpha(G)$. From the induction hypothesis we know that $f';g' \in \tau_\alpha(F;G)$ and from monotonicity of ; we get that $f;g \leq f';g'$. It follows that

$$f;g \in {\downarrow}\tau_\alpha(F;G) \subseteq \tau(\tau_\alpha(F;G)) = \tau_{\alpha+1}(F;G).$$

2. Case: $f \leq f'$ for some $f' \in \tau_\alpha(F)$ and $g = \sup_i g_i$ for an ω-chain $(g_i)_i$ in $\tau_\alpha(G)$. The induction hypothesis says that every $f';g_i$ is in $\tau_\alpha(F;G)$. Since ; is monotone, the sequence $(f';g_i)_i$ is an ω-chain in $\tau_\alpha(F;G)$. But $\mathcal{C}$ is an ω-continuous category (Definition 83), which implies that composition is ω-continuous in the right argument and hence $\sup_i f';g_i = f';\sup_i g_i = f';g$. So, we conclude that $f';g$ is in $\overline{\sup}(\tau_\alpha(F;G)) \subseteq \tau_{\alpha+1}(F;G)$.

3. Case: $f = \sup_i f_i$ for an ω-chain $(f_i)_i$ in $\tau_\alpha(F)$ and $g \leq g'$ for some $g' \in \tau_\alpha(G)$. The proof is analogous to the one given in the previous case, where now we use the fact that composition is ω-continuous in the first argument.

4. Case: $f = \sup_i f_i$ for an ω-chain $(f_i)_i$ in $\tau_\alpha(F)$ and $g = \sup_i g_i$ for an ω-chain $(g_i)_i$ in $\tau_\alpha(G)$. Composition is ω-continuous in both arguments, which gives us that

$$f;g = (\sup_i f_i);(\sup_j g_j) = \sup_i(f_i;\sup_j g_j) = \sup_i \sup_j f_i;g_j = \sup_i f_i;g_i.$$

From the induction hypothesis we get that every $f_i; g_i$ is in $\tau_\alpha(F; G)$, so $(f_i; g_i)_i$ is an ω-chain in $\tau_\alpha(F; G)$. We thus obtain that $f; g = \sup_i f_i; g_i$ is in $\overline{\sup}(\tau_\alpha(F; G)) \subseteq \tau_{\alpha+1}(F; G)$.

From the claim we have just proved, we easily obtain the desired property: $f \in \mathsf{cl}^{\mathfrak{I}}(F)$ and $g \in \mathsf{cl}^{\mathfrak{I}}(G)$ imply that $f; g \in \mathsf{cl}^{\mathfrak{I}}(F; G)$. The hypotheses say that $f \in \tau_\alpha(F)$ and $g \in \tau_\beta(G)$ for some ordinals α and β. Take any ordinal γ with $\alpha, \beta \leq \gamma$. It follows that $f \in \tau_\gamma(F)$ and $g \in \tau_\gamma(G)$ and by our claim we deduce that $f; g \in \tau_\gamma(F; G) \subseteq \mathsf{cl}^{\mathfrak{I}}(F; G)$.

We also have to prove that $\mathsf{cl}^{\mathfrak{I}}$ interacts well with the pairing operation $\langle \cdot, \cdot \rangle$. Since $\mathcal{C}$ is ω-continuous, we know that pairing is monotone and ω-continuous in both arguments. The proof proceeds exactly as in the case of composition. $\square$

Of central interest in the present work is the dagger † operation, which is interpreted in the standard **CPO** model as follows: $f^\dagger(y)$ is the least fixpoint of the mapping $f_y : x \mapsto f(x, y)$ for a **CPO**-function of type $f : X \times Y \to X$. More abstractly, we consider below in Definition 88 an axiomatically-defined class of ordered categories with a dagger operation that satisfies some typical properties of least fixpoints.

Definition 88 (Category With Parametric Least Fixpoints)**.** Let $\mathcal{C}$ be a pointed ordered category with products (Definition 70). We say that $\mathcal{C}$ has *parametric least fixpoints* if it has additionally a *dagger operation* † with the typing rule

$$\frac{f : X \times Y \to X}{f^\dagger : Y \to X}$$

that satisfies the following three universal Horn axioms [34]:

$$\langle f^\dagger, \mathsf{id}_Y\rangle; f \leq f^\dagger \text{ for } f : X \times Y \to X \tag{4.52}$$

$$\langle g, \mathsf{id}_Y\rangle; f \leq g \implies f^\dagger \leq g \text{ for } f : X \times Y \to X \text{ and } g : Y \to X \tag{4.53}$$

$$g; f^\dagger \leq [(\mathsf{id}_X \times g); f]^\dagger \text{ for } g : Z \to Y \text{ and } f : X \times Y \to X \tag{4.54}$$

The three axioms for † are called *pre-fixpoint inequality*, *least pre-fixpoint implication* or *Park induction rule*, and *parameter inequality* respectively.

In the observation below, we see that every ω-continuous category with binary products has parametric least fixpoints. In particular, the category **CPO** of ω-CPOs and ω-continuous functions (see Example 85) has parametric least fixpoints. As expected, because of ω-continuity least fixpoints can be defined in terms of suprema of countable chains.

Observation 89 (Least Fixpoints As Suprema of Countable Chains)**.** Let $\mathcal{C}$ be an arbitrary ω-continuous category with products. We define the parametric least fixpoint operation † as follows:

$$f^\dagger \triangleq \sup\nolimits_{n\geq 0} \upsilon_n \qquad \upsilon_0 \triangleq \bot_{YX} \qquad \upsilon_{n+1} \triangleq \langle \upsilon_n, \mathsf{id}_Y\rangle; f \tag{4.55}$$

for an arrow $f : X \times Y \to X$. A straightforward inductive argument establishes that the sequence υ_n is increasing. We show that the † operation satisfies the pre-fixpoint inequality:

$$\begin{aligned}\langle f^\dagger, \mathsf{id}_Y\rangle; f &= \langle \sup\nolimits_n \upsilon_n, \mathsf{id}_Y\rangle; f = (\sup\nolimits_n \langle \upsilon_n, \mathsf{id}_Y\rangle); f \\ &= \sup\nolimits_n \langle \upsilon_n, \mathsf{id}_Y\rangle; f = \sup\nolimits_n \upsilon_{n+1} \leq f^\dagger.\end{aligned}$$

For the least pre-fixpoint implication, assume that $\langle g, \mathsf{id}_Y\rangle; f \leq g$. We claim that $\upsilon_n \leq g$ for every $n \geq 0$. The base case $\bot_{YX} \leq g$ is trivial. For the induction step we have:

$$\begin{aligned} \upsilon_{n+1} &= \langle \upsilon_n, \mathsf{id}_Y \rangle ; f && \text{[definition of } \upsilon\text{]} \\ &\leq \langle g, \mathsf{id}_Y \rangle ; f && \text{[induction hypothesis, monotonicity of ; and } \langle \cdot, \cdot \rangle\text{]} \\ &\leq g. && \text{[assumption]} \end{aligned}$$

So, g is an upper bound of the sequence $(\upsilon_n)_n$ and therefore $f^\dagger = \sup_n \upsilon_n \leq g$. Finally, for the parameter inequality we have the definition:

$$[(\mathsf{id}_X \times g); f]^\dagger = \sup_n \upsilon'_n \quad \upsilon'_0 = \bot_{ZX} \quad \upsilon'_{n+1} = \langle \upsilon'_n, \mathsf{id}_Z \rangle ; (\mathsf{id}_X \times g); f = \langle \upsilon'_n, g \rangle ; f$$

We claim that $g; \upsilon_n = \upsilon'_n$ for every $n \geq 0$. Indeed, $g; \upsilon_0 = g; \bot_{YX} = \bot_{ZX} = \upsilon'_0$ for the base case, and for induction step we have:

$$g; \upsilon_{n+1} = g; \langle \upsilon_n, \mathsf{id}_Y \rangle ; f = \langle g; \upsilon_n, g \rangle ; f = \langle \upsilon'_n, g \rangle ; f = \upsilon'_{n+1}.$$

So, we conclude that $g; f^\dagger = g; (\sup_n \upsilon_n) = \sup_n g; \upsilon_n = \sup_n \upsilon'_n = [(\mathsf{id}_X \times g); f]^\dagger$.

The embedding theorem that follows (Theorem 90) is a variation of the previous Theorem 81 that is specifically about the subclass of ω-continuous categories. We use here the more elaborate construction of ideals (instead of lowersets) so that the induced nondeterministic category satisfies an additional property for the preservation of determinism. This will turn out to be crucial for the central result of this chapter, which is the (proof-theoretic) conservative extension of the theory of † in the standard **CPO** model. Theorem 90 also introduces the definition of dagger in terms of star in nondeterministic models (Equation 4.57).

Theorem 90 (Second Embedding Theorem)**.** Let $\mathcal{C}$ be an ω-continuous category with products (Definition 83), and consider the family $\mathsf{cl}^{\mathfrak{I}}_{XY}$ of ideal-closure operators for every homset $\mathcal{C}(X, Y)$ described in Definition 86. The universal Horn implication

$$D(\phi) \wedge D(\psi) \wedge \psi \leq \psi; \phi \implies D(\psi; \phi^*) \text{ for } \phi : Y \rightsquigarrow Y \text{ and } \psi : X \rightsquigarrow Y \quad (4.56)$$

is true in the nondeterministic category $\mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}$ (recall Def. 78). Moreover, the map

$$\mathsf{cl}^{\mathfrak{I}} : f \in \mathcal{C}(X,Y) \mapsto \mathsf{cl}^{\mathfrak{I}}_{XY}(f) \in \mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}(X,Y)$$

preserves the operation of parametric least fixpoints, which is defined in $\mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}$ as

$$\phi^{\dagger} \triangleq \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle; \langle\!\langle \phi, \varpi_2 \rangle\!\rangle^{*}; \varpi_1 \text{ for } \phi \in \mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}(X \times Y, X). \tag{4.57}$$

In fact, the mapping $\mathsf{cl}^{\mathfrak{I}}$ is an order-preserving injective homomorphism (embedding) whose image is the deterministic subcategory of $\mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}$.

Proof. First, let us observe that the category $\mathcal{C}$ and the family $\mathsf{cl}^{\mathfrak{I}}$ of closure operations satisfy the conditions of Theorem 81. So, the category $\mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}$ (see Definition 78) is a nondeterministic category with products and joins. Moreover, we already know that $\mathsf{cl}^{\mathfrak{I}}$ is order-preserving and commutes with all operations except for dagger, which we have to consider here.

To establish the truth of Axiom (4.56) in $\mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}$, we consider arbitrary morphisms $f \in \mathcal{C}(Y,Y)$ and $g \in \mathcal{C}(X,Y)$, and we make the assumption that

$$\mathsf{cl}^{\mathfrak{I}}(g) \subseteq \mathsf{cl}^{\mathfrak{I}}(g); \mathsf{cl}^{\mathfrak{I}}(f) = \mathsf{cl}^{\mathfrak{I}}(g; f).$$

We want to show that the following element of the category $\mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}$ is deterministic:

$$\begin{aligned}
\mathsf{cl}^{\mathfrak{I}}(g); \mathsf{cl}^{\mathfrak{I}}(f)^{*} &= \mathsf{cl}^{\mathfrak{I}}(g); \textstyle\sum_{n \geq 0} \mathsf{cl}^{\mathfrak{I}}(f)^{n} && [\text{definition of } {}^{*} \text{ in } \mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}] \\
&= \textstyle\sum_{n \geq 0} \mathsf{cl}^{\mathfrak{I}}(g); \mathsf{cl}^{\mathfrak{I}}(f)^{n} && [\text{Theorem 81, Axiom (4.35)}] \\
&= \textstyle\sum_{n \geq 0} \mathsf{cl}^{\mathfrak{I}}(g; f^{n}) && [\text{Theorem 81, } \mathsf{cl}^{\mathfrak{I}} \text{ homomorphism}] \\
&= \mathsf{cl}^{\mathfrak{I}}(\textstyle\bigcup_{n \geq 0} \mathsf{cl}^{\mathfrak{I}}(g; f^{n})) && [\text{definition of } \textstyle\sum \text{ in } \mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}] \\
&= \mathsf{cl}^{\mathfrak{I}}(\{g; f^{n} \mid n \geq 0\}) && [\text{Equation (4.45)}]
\end{aligned}$$

From our hypothesis $\mathsf{cl}^{\mathfrak{I}}(g) = {\downarrow} g \subseteq {\downarrow}(g; f) = \mathsf{cl}^{\mathfrak{I}}(g; f)$ we deduce that $g \leq g; f$. It follows that the sequence $(g; f^{n})_{n \geq 0}$ is an ω-chain in $\mathcal{C}(X,Y)$. So, we have that

$$\mathsf{cl}^{\mathfrak{I}}(g); \mathsf{cl}^{\mathfrak{I}}(f)^{*} = \mathsf{cl}^{\mathfrak{I}}(\{g; f^{n} \mid n \geq 0\}) = {\downarrow}(\sup_{n \geq 0} g; f^{n}) = \mathsf{cl}^{\mathfrak{I}}(\sup_{n \geq 0} g; f^{n}),$$

which proves the desired conclusion that $\mathsf{cl}^{\mathfrak{I}}(g);\mathsf{cl}^{\mathfrak{I}}(f)^{*}$ is deterministic.

For the second part of the theorem, we only need to prove that the map $\mathsf{cl}^{\mathfrak{I}}$ commutes with the parametric least fixpoints operation given the results of Theorem 81. Consider an arbitrary element $f : X \times Y \to X$ of $\mathcal{C}$ and recall the definitions

$$f^{\dagger} = \sup_{n \geq 0} \upsilon_n \qquad \upsilon_0 = \bot_{YX}$$
$$\mathsf{cl}^{\mathfrak{I}}(f)^{\dagger} = \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{*} ; \varpi_1 \qquad \upsilon_{n+1} = \langle \upsilon_n, \mathsf{id}_Y \rangle ; f$$

We want to show that $\mathsf{cl}^{\mathfrak{I}}(f^{\dagger}) = \mathsf{cl}^{\mathfrak{I}}(f)^{\dagger}$. For this, we first establish the claim that

$$\langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{n} = \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\upsilon_n), \eta_Y \rangle\!\rangle$$

for every $n \geq 0$. For the base case $n = 0$, we simply observe that:

$$\langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{0} = \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \eta_{X \times Y} = \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle$$
$$\langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\upsilon_0), \eta_Y \rangle\!\rangle = \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\bot_{YX}), \eta_Y \rangle\!\rangle = \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle$$

For the induction step, we make use of the induction hypothesis and several axioms from Definition 79:

$$\begin{aligned}
&\langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{n+1} = && \text{[definition of } f^{n+1}\text{]}\\
&\langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{n} ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle = && \text{[induction hypothesis]}\\
&\langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\upsilon_n), \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle = && \text{[Equation 4.40 in Claim 80]}\\
&\langle\!\langle \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\upsilon_n), \eta_Y \rangle\!\rangle ; \mathsf{cl}^{\mathfrak{I}}(f), \eta_Y \rangle\!\rangle = && \text{[Theorem 81, } \mathsf{cl}^{\mathfrak{I}} \text{ homomorphism]}\\
&\langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\langle \upsilon_n, \mathsf{id}_Y \rangle ; f), \eta_Y \rangle\!\rangle = && \text{[definition of } \upsilon_{n+1}\text{]}\\
&\langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\upsilon_{n+1}), \eta_Y \rangle\!\rangle .
\end{aligned}$$

Finally, we use the claim we have just proved and properties of nondeterministic categories:

$$\begin{aligned}
\mathsf{cl}^{\mathfrak{I}}(f)^{\dagger} &= \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{*} ; \varpi_1 && \text{[definition of } ^{\dagger} \text{ in } \mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}] \\
&= \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \left(\textstyle\sum_{n\geq 0} \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{n} \right) ; \varpi_1 && \text{[definition of } ^{*} \text{ in } \mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}] \\
&= \textstyle\sum_{n\geq 0} \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(f), \varpi_2 \rangle\!\rangle^{n} ; \varpi_1 && \text{[distributivity]} \\
&= \textstyle\sum_{n\geq 0} \langle\!\langle \mathsf{cl}^{\mathfrak{I}}(\upsilon_n), \eta_Y \rangle\!\rangle ; \varpi_1 && \text{[previous claim]} \\
&= \textstyle\sum_{n\geq 0} \mathsf{cl}^{\mathfrak{I}}(\upsilon_n) && \text{[projection]} \\
&= \mathsf{cl}^{\mathfrak{I}}(\textstyle\bigcup_{n\geq 0} \mathsf{cl}^{\mathfrak{I}}(\upsilon_n)) && \text{[definition of } \textstyle\sum \text{ in } \mathcal{C}_{\mathsf{cl}^{\mathfrak{I}}}] \\
&= \mathsf{cl}^{\mathfrak{I}}(\{\upsilon_n \mid n \geq 0\}), && \text{[Equation (4.45)]}
\end{aligned}$$

which is equal to $\downarrow \sup_{n\geq 0} \upsilon_n = \downarrow f^{\dagger} = \mathsf{cl}^{\mathfrak{I}}(f^{\dagger})$. So, $\mathsf{cl}^{\mathfrak{I}}$ commutes with the dagger operation. □

Observation 91 (Parametric Least Fixpoints and Determinism)**.** Let $\mathcal{C}$ be a nondeterministic category with products and joins that satisfies additionally the universal Horn implication

$$D(\phi) \wedge D(\psi) \wedge \psi \leq \psi ; \phi \implies D(\psi ; \phi^{*}) \text{ for } \phi : Y \rightsquigarrow Y \text{ and } \psi : X \rightsquigarrow Y.$$

An immediate consequence is that *parametric least fixpoints preserve determinism*, that is,

$$D(\phi) \implies D(\phi^{\dagger}) \text{ for } \phi : X \times Y \rightsquigarrow X$$

is derivable, where † is defined from * as in Equation (4.57).

Proof. We assume that ϕ is deterministic. Since ϖ_1 is deterministic and ; preserves determinism, it suffices to show that the arrow $\langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \phi, \varpi_2 \rangle\!\rangle^{*}$ is deterministic. By virtue of the extra axiom we have stipulated, we need to show:

$$\begin{aligned}\langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \langle\!\langle \phi, \varpi_2 \rangle\!\rangle &= \langle\!\langle \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \phi, \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \varpi_2 \rangle\!\rangle \\ &= \langle\!\langle \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle ; \phi, \eta_Y \rangle\!\rangle \\ &\geq \langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle .\end{aligned}$$

The first equality above is justified by the fact that $\langle\!\langle \perp\!\!\!\perp_{YX}, \eta_Y \rangle\!\rangle$ is deterministic.

□

The results of this section have brought us a lot closer to our final goal of showing that we can reason about fixpoints using properties of nondeterminism and star. We have shown that for all ω-continuous categories with products there is an *ideal-closure* construction that enriches the categories with nondeterministic structure. This construction is a conservative extension that allows us to define parametric fixpoints using star.

4.4 Typed Kleene Algebra With Products

In this section we propose the central axiomatization of typed Kleene algebras with products. It is a *finitary* system consisting of Horn implications that involve the equality predicate as well as a *deterministic subtype* unary predicate that distinguishes the deterministic elements. This axiomatization intends to capture several key properties of nondeterministic categories (recall the infinitary axiomatization of Definition 79) in a weaker system. For this we include the crucial axioms for * of Kleene algebras [69, 70], as well several additional axioms for the interaction between products and nondeterminism. Our main result here is that these Kleene algebras with products possess sufficient structure to define

product	$(X,Y) \mapsto X \times Y$	left projection	$\pi_1^{XY} : X \times Y \rightsquigarrow X$
identity	$\mathsf{id}_X : X \rightsquigarrow X$	right projection	$\pi_2^{XY} : X \times Y \rightsquigarrow Y$
bottom	$\perp_{XY} : X \rightsquigarrow Y$		

composition $$\frac{f : X \rightsquigarrow Y \qquad g : Y \rightsquigarrow Z}{f; g : X \rightsquigarrow Z}$$

pairing $$\frac{f : X \rightsquigarrow Y \qquad g : X \rightsquigarrow Z}{\langle f, g \rangle : X \rightsquigarrow Y \times Z}$$

product $$\frac{f : X \rightsquigarrow Y \qquad g : X' \rightsquigarrow Y'}{f \times g \triangleq \langle \pi_1; f, \pi_2; g \rangle : X \times X' \rightsquigarrow Y \times Y'}$$

nondeterministic choice $$\frac{f : X \rightsquigarrow Y \qquad g : X \rightsquigarrow Y}{f + g : X \rightsquigarrow Y}$$

nondeterministic iteration $$\frac{f : X \rightsquigarrow X}{f^* : X \rightsquigarrow X}$$

dagger $$\frac{f : X \times Y \rightsquigarrow X}{f^\dagger \triangleq \langle \perp_{YX}, \mathsf{id}_Y \rangle; \langle f, \pi_2 \rangle^*; \pi_1 : Y \rightsquigarrow X}$$

Figure 4.3: Constants and operations for typed Kleene algebras with products.

a parametric least fixpoints operator † from * that satisfies the desired properties of Definition 88. We stress that the defined † operator extends to all elements of the appropriate type, not just the deterministic ones.

Definition 92 (KA With Products)**.** A *typed Kleene algebra $\mathcal{K}$ with (binary) products* is a typed algebraic structure with the operations of Figure 4.3 and a unary *determinism* predicate D on every homset such that $\mathcal{K}$ is a model of the following universal Horn axioms:

$$D(\mathsf{id}_X) \qquad D(\pi_1^{XY}) \qquad D(\pi_2^{XY}) \qquad D(\perp_{XY}) \tag{4.58}$$

$$D(f) \wedge D(g) \implies D(f; g) \text{ for } f : X \rightsquigarrow Y \text{ and } g : Y \rightsquigarrow Z \tag{4.59}$$

$$D(f) \wedge D(g) \implies D(\langle f, g \rangle) \text{ for } f : X \rightsquigarrow Y \text{ and } g : X \rightsquigarrow Z \tag{4.60}$$

$$(f; g); h = f; (g; h) \text{ for } f : X \rightsquigarrow Y,\ g : Y \rightsquigarrow Z \text{ and } h : Z \rightsquigarrow W \tag{4.61}$$

$$\mathsf{id}_X; f = f \text{ for } f : X \rightsquigarrow Y \tag{4.62}$$

$$f;\mathsf{id}_Y = f \text{ for } f : X \rightsquigarrow Y \tag{4.63}$$

$$\langle f, g\rangle;\pi_1 = f \text{ for } f : X \rightsquigarrow Y \text{ and } g : X \rightsquigarrow Z \tag{4.64}$$

$$\langle f, g\rangle;\pi_2 = g \text{ for } f : X \rightsquigarrow Y \text{ and } g : X \rightsquigarrow Z \tag{4.65}$$

$$\langle h;\pi_1, h;\pi_2\rangle \geq h \text{ for } h : X \rightsquigarrow Y \times Z \tag{4.66}$$

$$D(h) \implies \langle h;\pi_1, h;\pi_2\rangle = h \text{ for } h : X \rightsquigarrow Y \times Z \tag{4.67}$$

$$D(f) \implies f;\langle g_1, g_2\rangle = \langle f;g_1, f;g_2\rangle \tag{4.68}$$
$$\text{for } f : X \rightsquigarrow Y \text{ and } g_i : Y \rightsquigarrow Z_i$$

$$\langle f_1, f_2\rangle;(g_1 \times g_2) = \langle f_1;g_1, f_2;g_2\rangle \text{ for } f_i : X \rightsquigarrow Y_i \text{ and } g_i : Y_i \rightsquigarrow Z_i \tag{4.69}$$

$$\langle f, g\rangle;\langle \pi_2, \pi_1\rangle = \langle g, f\rangle \text{ for } f : X \rightsquigarrow Y \text{ and } g : X \rightsquigarrow Z \tag{4.70}$$

$$\langle f, \pi_2\rangle;\langle g, \pi_2\rangle = \langle\langle f, \pi_2\rangle;g, \pi_2\rangle \text{ for } f, g : X \times Y \rightsquigarrow X \tag{4.71}$$

$$\bot_{XY} \leq f \text{ for } f : X \rightsquigarrow Y \tag{4.72}$$

$$f;\bot_{YZ} = \bot_{XZ} \text{ for } f : X \rightsquigarrow Y \tag{4.73}$$

$$\langle \bot_{XY}, \bot_{XZ}\rangle = \bot_{X,Y\times Z} \tag{4.74}$$

$$(f + g) + h = f + (g + h) \text{ for } f, g, h : X \rightsquigarrow Y \tag{4.75}$$

$$f + g = g + f \text{ for } f, g : X \rightsquigarrow Y \tag{4.76}$$

$$f + f = f \text{ for } f : X \rightsquigarrow Y \tag{4.77}$$

$$(f_1 + f_2);g = f_1;g + f_2;g \text{ for } f_i : X \rightsquigarrow Y \text{ and } g : Y \rightsquigarrow Z \tag{4.78}$$

$$f;(g_1 + g_2) = f;g_1 + f;g_2 \text{ for } f : X \rightsquigarrow Y \text{ and } g_i : Y \rightsquigarrow Z \tag{4.79}$$

$$\langle f_1 + f_2, g\rangle = \langle f_1, g\rangle + \langle f_2, g\rangle \text{ for } f_i : X \rightsquigarrow Y \text{ and } g : X \rightsquigarrow Z \tag{4.80}$$

$$\langle f, g_1 + g_2\rangle = \langle f, g_1\rangle + \langle f, g_2\rangle \text{ for } f : X \rightsquigarrow Y \text{ and } g_i : X \rightsquigarrow Z \tag{4.81}$$

$$\mathsf{id}_X + f;f^* \leq f^* \text{ for } f : X \rightsquigarrow X \tag{4.82}$$

$$\mathsf{id}_X + f^*;f \leq f^* \text{ for } f : X \rightsquigarrow X \tag{4.83}$$

$$f;g \leq g \implies f^*;g \leq g \text{ for } f : X \rightsquigarrow X \text{ and } g : X \rightsquigarrow Y \tag{4.84}$$

$$g;f \leq g \implies g;f^* \leq g \text{ for } g : X \rightsquigarrow Y \text{ and } f : Y \rightsquigarrow Y \tag{4.85}$$

$$\langle f, \pi_2 \rangle^* = \langle \langle f, \pi_2 \rangle^*; \pi_1, \langle f, \pi_2 \rangle^*; \pi_2 \rangle \text{ for } f : X \times Y \rightsquigarrow X \tag{4.86}$$

where the partial order $\leq$ is induced by $+$ as follows: $f \leq g$ iff $f + g = g$. We also consider a *dagger* operation that is defined from * as follows:

$$f^\dagger \triangleq \langle \bot_{YX}, \mathsf{id}_Y \rangle; \langle f, \pi_2 \rangle^*; \pi_1 \text{ for } f : X \times Y \rightsquigarrow X. \tag{4.87}$$

We remark for Equation (4.86) that it is some kind of extension of the pairing-uniqueness axiom to nondeterministic elements of a special form.

The theorem that follows says that our axiomatization of Kleene algebras with products is strong enough to entail the desired properties of the defined † operation. The standard four axioms for * as well as Equation (4.86) for the interaction between * and products play a crucial role in the proof.

Theorem 93 (Parametric Least Fixpoints in KA)**.** Let $\mathcal{K}$ be a typed KA with products. The defined dagger operation † of Equation (4.87) satisfies the axioms of parametric least fixpoints (recall Definition 88):

$$\langle f^\dagger, \mathsf{id}_Y \rangle; f \leq f^\dagger \text{ for } f : X \times Y \rightsquigarrow X$$

$$\langle g, \mathsf{id}_Y \rangle; f \leq g \implies f^\dagger \leq g \text{ for } f : X \times Y \rightsquigarrow X \text{ and } g : Y \rightsquigarrow X$$

$$g; f^\dagger \leq [(\mathsf{id}_X \times g); f]^\dagger \text{ for } g : Z \rightsquigarrow Y \text{ and } f : X \times Y \rightsquigarrow X$$

Proof. Throughout the proof we write f to range over an arbitrary element of type $X \times Y \rightsquigarrow X$.

First, we will show the equation $\langle f, \pi_2 \rangle^*; \pi_2 = \pi_2$. Since $\mathsf{id}_{X \times Y} \leq \langle f, \pi_2 \rangle^*$ and ; is monotone, we obtain that $\pi_2 \leq \langle f, \pi_2 \rangle^*; \pi_2$. It remains to see that $\langle f, \pi_2 \rangle^*; \pi_2 \leq \pi_2$, which is implied by $\langle f, \pi_2 \rangle; \pi_2 \leq \pi_2$. The last inequality is true because $\pi_2 \leq \pi_2$.

Now, we are ready to prove the first property $\langle f^\dagger, \mathsf{id}_Y\rangle; f \leq f^\dagger$ for the dagger operation. Using the claim of the two previous paragraph we see that:

$$\begin{aligned}
f^\dagger &= \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^*; \pi_1 && \text{[definition of }^\dagger\text{]} \\
&= \langle \bot_{YX}, \mathsf{id}_Y\rangle; (1 + \langle f, \pi_2\rangle^*; \langle f, \pi_2\rangle); \pi_1 && \text{[unfold }^*\text{]} \\
&= \langle \bot_{YX}, \mathsf{id}_Y\rangle; (\pi_1 + \langle f, \pi_2\rangle^*; f) && \text{[right distributivity]} \\
&= \bot_{YX} + \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^*; f && \text{[left distributivity]} \\
&= \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^*; f && \text{[Axiom (4.72)]} \\
\langle f^\dagger, \mathsf{id}_Y\rangle &= \langle \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^*; \pi_1, \mathsf{id}_Y\rangle && \text{[definition of }^\dagger\text{]} \\
&= \langle \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^*; \pi_1, \langle \bot_{YX}, \mathsf{id}_Y\rangle; \pi_2\rangle && \text{[projection]} \\
&= \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle \langle f, \pi_2\rangle^*; \pi_1, \pi_2\rangle && \text{[Axiom (4.68)]} \\
&= \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle \langle f, \pi_2\rangle^*; \pi_1, \langle f, \pi_2\rangle^*; \pi_2\rangle && \text{[previous claim]} \\
&= \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^* && \text{[Axiom (4.86)]}
\end{aligned}$$

and therefore $\langle f^\dagger, \mathsf{id}_Y\rangle; f = \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^*; f = f^\dagger$. For the least pre-fixpoint implication we suppose that $\langle g, \mathsf{id}_Y\rangle; f \leq g$ where $g : Y \rightsquigarrow X$. We claim that:

$$\begin{aligned}
&\langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^* \leq \langle g, \mathsf{id}_Y\rangle \Longleftarrow \\
&\langle \bot_{YX}, \mathsf{id}_Y\rangle \leq \langle g, \mathsf{id}_Y\rangle \text{ and } \langle g, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle \leq \langle g, \mathsf{id}_Y\rangle \overset{\bot \leq g}{\Longleftarrow} \\
&\langle g, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle \leq \langle \langle g, \mathsf{id}_Y\rangle; f, \langle g, \mathsf{id}_Y\rangle; \pi_2\rangle = \langle \langle g, \mathsf{id}_Y\rangle; f, \mathsf{id}_Y\rangle \leq \langle g, \mathsf{id}_Y\rangle,
\end{aligned}$$

which holds because of our assumption that $\langle g, \mathsf{id}_Y\rangle; f \leq g$. Finally, we consider an arbitrary element $g : Z \rightsquigarrow Y$. To show the inequality $g; f^\dagger \leq [(\mathsf{id}_X \times g); f]^\dagger$ is suffices to establish

$$\begin{aligned}
&g; \langle \bot_{YX}, \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^* \leq && \text{[Axiom (4.66)]} \\
&\langle g; \bot_{YX}, g; \mathsf{id}_Y\rangle; \langle f, \pi_2\rangle^* = && \text{[Axiom (4.73)]} \\
&\langle \bot_{ZX}, g\rangle; \langle f, \pi_2\rangle^* \leq && \text{[to show]} \\
&\langle \bot_{ZX}, \mathsf{id}_Z\rangle; \langle (\mathsf{id}_X \times g); f, \pi_2\rangle^*; (\mathsf{id}_X \times g),
\end{aligned}$$

which is implied by $\langle \bot_{ZX}, g\rangle \leq \langle \bot_{ZX}, \mathsf{id}_Z\rangle; (\mathsf{id}_X \times g) = \langle \bot_{ZX}, g\rangle$ and

$$\begin{array}{ll}
\langle \bot_{ZX}, \mathsf{id}_Z\rangle; \langle (\mathsf{id}_X \times g); f, \pi_2\rangle^*; (\mathsf{id}_X \times g); \langle f, \pi_2\rangle \leq & \text{[Axiom (4.66)]} \\
\langle \bot_{ZX}, \mathsf{id}_Z\rangle; \langle (\mathsf{id}_X \times g); f, \pi_2\rangle^*; \langle (\mathsf{id}_X \times g); f, (\mathsf{id}_X \times g); \pi_2\rangle = & \text{[projection]} \\
\langle \bot_{ZX}, \mathsf{id}_Z\rangle; \langle (\mathsf{id}_X \times g); f, \pi_2\rangle^*; \langle (\mathsf{id}_X \times g); f, \pi_2; g\rangle = & \text{[Axiom (4.69)]} \\
\langle \bot_{ZX}, \mathsf{id}_Z\rangle; \langle (\mathsf{id}_X \times g); f, \pi_2\rangle^*; \langle (\mathsf{id}_X \times g); f, \pi_2\rangle; (\mathsf{id}_X \times g) \leq & [x^*; x \leq x^*] \\
\langle \bot_{ZX}, \mathsf{id}_Z\rangle; \langle (\mathsf{id}_X \times g); f, \pi_2\rangle^*; (\mathsf{id}_X \times g). &
\end{array}$$

We have thus established all three properties of parametric least fixpoints for arbitrary nondeterministic elements of type $X \times Y \rightsquigarrow X$ using only the axioms of typed KA with products. □

4.5 The Theory of Parametric Fixpoints

We have already developed in the previous sections all the technical machinery we need to establish proof-theoretic conservativity results for the *equational theory of* † (denoted $\mathsf{IT}^=$), as well as for the *theory of valid inequalities for* † (denoted $\mathsf{IT}^\leq$). These results essentially say that can reason about parametric fixpoints using the language of regular expressions with products and an appropriate typed variant of KA. So, the * operation of Kleene algebra, which can be characterized in terms of least fixpoints, should not be considered as being merely a special case of fixpoints. Our results show that * is versatile enough to allow the faithful encoding of general parametric fixpoints.

First, consider the typed algebraic language of categories with binary products and parametric least fixpoints. We fix a set Ω of *base types*. We write $A, B, \ldots$ to range over these base types. The set $\mathsf{Types}(\Omega)$ of all *types* over Ω is given by the following grammar:

$$X, Y ::= \text{base type } A \in \Omega \mid X \times Y.$$

We use letters $X, Y, Z, \ldots$ to range over arbitrary types. A *typed term* is an expression of the form $f : X \to Y$, where f is a term and X, Y are types in $\mathsf{Types}(\Omega)$. We also say that $X \to Y$ is the type of f. The constant symbols and constructors for typed terms are the following:

$$\mathsf{id}_X : X \to X \quad \pi_1^{XY} : X \times Y \to X \quad \pi_2^{XY} : X \times Y \to Y \quad \bot_{XY} : X \to Y$$

$$\frac{f : X \to Y \qquad g : Y \to Z}{f;g : X \to Z} \quad \frac{f : X \to Y \qquad g : X \to Z}{\langle f, g\rangle : X \to Y \times Z} \quad \frac{f : X \times Y \to X}{f^\dagger : Y \to X}$$

As usual, we assume that we have a countable supply of variables of each type.

For the typed language of KA with products, we write $f : X \rightsquigarrow Y$ for an arbitrary typed term. The constants and constructors for typed KA terms are as in the previous paragraph, except that the rule for † is replaced by the rules

$$\frac{f : X \rightsquigarrow Y \qquad g : X \rightsquigarrow Y}{f + g : X \rightsquigarrow Y} \qquad\qquad \frac{f : X \rightsquigarrow X}{f^* : X \rightsquigarrow X}$$

for nondeterministic choice $+$ and nondeterministic iteration *.

Definition 94 (Translation)**.** We define the translation $[\cdot]$ from the language of categories with † to the language of KA with products. All variables and constants remain the same. For example, $[u : X \to Y] = u : X \rightsquigarrow Y$ for a variable, and $[\pi_1 : X \times Y \to X] = \pi_1 : X \times Y \rightsquigarrow X$ for the left projection constant. The dagger is translated as

$$[f^\dagger] = \langle \bot_{YX}, \mathsf{id}_Y\rangle ; \langle [f], \pi_2\rangle^* ; \pi_1.$$

The translation function $[\cdot]$ commutes with the rest of the operation symbols.

For the rest of this section, we will use the term *extended KA (with products)* to refer to the axiomatization that extends the one for Kleene algebras with products (Definition 92) with

$$D(f) \wedge D(g) \wedge g \leq g; f \implies D(g; f^*) \text{ for } f : Y \rightsquigarrow Y \text{ and } g : X \rightsquigarrow Y. \quad (4.88)$$

The above implication is an additional property about the preservation of determinism. We will write $\mathsf{KA}^\times$ to denote the equations that are provable in extended KA. Moreover, we write $\mathsf{IT}^=$ for the equations $f = g$ that are valid in **CPO**, and $\mathsf{IT}^\leq$ for the valid inequalities $f \leq g$.

Theorem 95 ($\mathsf{KA}^\times$ Conservatively Extends $\mathsf{IT}^\leq$)**.** For an inequality $f \leq g$ in the language of categories with †, we have that $\mathbf{CPO} \models f \leq g$ iff $[f] \leq [g]$ is a theorem of $\mathsf{KA}^\times$.

Proof. First, we observe that for every term f in the language of $\mathsf{IT}^\leq$, we can show in extended KA that $D([f])$, which says that $[f]$ is deterministic. This is because all the constants id, π_1, π_2, and $\perp$ are deterministic, the operations ; and $\langle \cdot, \cdot \rangle$ preserve determinism, and the additional axiom implies that † also preserves determinism (see Observation 91).

For the completeness part of the theorem, we assume that $\mathbf{CPO} \models f \leq g$ and we show that $[f] \leq [g]$ is in $\mathsf{KA}^\times$. By previous results of Ésik in [34], we know that the axiomatization of categories with parametric least fixpoints (Definition 88) is complete for the theory of **CPO**. Since the translations of typed terms are all provably deterministic, it suffices to observe that all (translations of) the axioms of Definition 70 and 88 are provable in extended KA. Most of the work for this has been already done in Theorem 93.

For the soundness part, suppose that $[f] \leq [g]$ is provable in extended KA. Since **CPO** is an ω-continuous category (Definition 83), the category $\mathbf{CPO}_{\mathsf{cl}\mathfrak{I}}$ that arises from the ideal-closure construction satisfies all axioms of extended

KA by virtue of Theorem 90. It follows that $\mathbf{CPO}_{\mathsf{cl}\mathfrak{J}} \models [f] \leq [g]$. But both $[f]$ and $[g]$ denote deterministic elements of $\mathbf{CPO}_{\mathsf{cl}\mathfrak{J}}$, and we also know from Theorem 90 that the deterministic part of $\mathbf{CPO}_{\mathsf{cl}\mathfrak{J}}$ is isomorphic to $\mathbf{CPO}$. We thus conclude that $\mathbf{CPO} \models f \leq g$. □

Corollary 96 ($\mathsf{KA}^{\times}$ Conservatively Extends $\mathsf{IT}^{=}$)**.** For an equation $f = g$ in the language of categories with †, we have that $\mathbf{CPO} \models f = g$ iff $[f] = [g]$ is a theorem of $\mathsf{KA}^{\times}$.

Proof. This is an immediate consequence of Theorem 95. □

We have presented above two *syntactic conservativity results*, which were obtained by considering the categories satisfying the so called "Park axiomatization" of Definition 88. Even though our model-theoretic constructions do not apply to all models of $\mathsf{IT}^{=}$ (the categories that are called *iteration theories* by Bloom and Ésik) or $\mathsf{IT}^{\leq}$ (the *ordered iteration theories*), we have shown that our typed variant of KA captures faithfully the basic syntactic theories of parametric fixpoints.

4.6 Related Work

There is a long line of work, primarily by Bloom and Ésik, under the name of "iteration theories" or the "(in)equational theory of iteration" (see e.g. [17, 19, 20, 34, 120] and references therein), which is intimately related to the work on Kleene algebra [31, 68, 69, 70, 72, 62, 81, 74] in general and the present work in particular. The axioms of iteration theories capture the equational properties of fixpoints in several classes of structures relevant to computer science.

For example, they capture the equational theory of ω-continuous functions between ω-CPOs, where the algebraic signature includes symbols for composition, pairing, and parametric fixpoints. Several different infinite equational axiomatizations have been considered in the literature, all of which require substantial effort to parse and understand. By allowing quasi-equations, much simpler (finite) axiomatizations can be found. Many examples of iteration theories involve functions on posets, so it is a natural question to look for complete axiomatizations of the valid inequalities over classes of structures that are of interest, e.g., structures of ω-continuous functions over ω-CPOs. One such universal Horn axiomatization is given in [34]. This axiomatization includes two inequalities and one implication for the † operation, which are both intuitive and easy to memorize. We note that in the work on iteration theories, the issue of how (non)determinism interacts with (non-strict) pairs, which is central in the present work, is handled in a way that excludes many natural models.

Of particular relevance to the relationship between iteration theories and Kleene algebra are the works on the so-called *matrix iteration theories* [18, 17, 20]. They are cartesian categories in which the homsets are commutative monoids with respect to an operation $+$, which distributes over composition. This also induces cocartesian structure and allows an easy translation between the dagger (parametric fixpoint) operation and Kleene star. However, this translation is not sound for the classes of structures we consider. In particular, the $+$ symbol cannot be generally understood as nondeterministic choice when $\langle \cdot, \cdot \rangle$ is interpreted as pairing: the axioms imply the property $\langle a, \bot \rangle + \langle \bot, b \rangle = \langle a, b \rangle$, which is not meaningful for nondeterministic computation with pairs. A program that nondeterministically returns either a pair $\langle a, \bot \rangle$ (right component diverging) or a pair $\langle \bot, b \rangle$ (left component diverging) is not equal to the program that returns the pair $\langle a, b\rangle$ (both components non-diverging). The translation of the dagger operation in the language of KA that we give here is crucially different, and is in fact sound for the class of matrix iteration theories as well. The star-to-dagger translation of [18] can be recovered as a simple degenerate case in our framework where all arrows are deterministic.

The work of Plotkin and Power on algebraic operations (see e.g. [105] for an overview) provides a uniform semantics of computational effects by considering primitive operations of type $f_X : (PX)^n \to PX$, where P is a monad, that are the source of effects. Their framework can be instantiated with a monad for nondeterminism, but their investigations are very general and they do not offer any technical machinery that can be of help here.

There is a somewhat tenuous connection with the work by Goncharov [43], who also studies some kind of interaction between nondeterminism and pairs in an algebraic/categorical setting. He defines *additive (strong) monads* and *Kleene monads* axiomatically. Calculi for an extended metalanguage of effects are defined and completeness/incompleteness results are obtained. Our notions of *nondeterministic categories* and *typed KAs with products* are different from that of a Kleene monad in that we consider non-strict programs that form lazy pairs. The absence of the strictness axiom $\bot; f = \bot$ from our axiomatization and the use of lazy pairs are essential for our encoding of fixpoints. In particular, the axioms stipulated in [43] would force all the parametric fixpoints to be equal to $\bot$, because in that system

$$\langle \bot, \mathsf{id}\rangle; \langle f, \pi_2\rangle^*; \pi_1 = \bot; \langle f, \pi_2\rangle^*; \pi_1 = \bot.$$

So, the models we are investigating in the present work are crucially different from the models considered in [43], in which parametric fixpoints trivialize.

The work on Hoare powerdomains [1, 104], which give models of angelic nondeterministic computations, is related. The (lower) Hoare powerdomain of a domain is formed by taking all the ideals of the domain, where by *ideal* we mean here the nonempty Scott-closed (down-closed and closed under suprema of directed subsets) subsets of the domain. In the present work, we identify models of the axiomatically defined *nondeterministic categories* which are somewhat similar to (and much simpler than) the construction of Hoare powerdomains over ω-CPOs. We first identify a simple model that arises from lowerset-closure operators on the category of posets with bottom elements. Then, we also prove that the ideal-closure operators on the category of ω-CPOs give rise to a model.

Our work here builds directly upon the existing work on Kleene algebra [31, 68, 69, 70, 72]. The crucial axioms for the iteration operation * are taken from [69, 70]. The system of KA we present is a typed Kleene algebra in the sense of [74] extended with products that satisfy weaker axioms than those of categorical products.

4.7 Conclusion

In this chapter we have reconciled the notions of iteration captured by the star operation * of KA and the dagger operation † of IT. We have presented and investigated a system of typed KA with products, in which the notion of a deterministic program turns out to be of importance. We work in the framework of categories with products combined with extra structure to treat (angelic) nondeterminism. We have identified two concrete models that arise from lowerset-closure operators on pointed posets and ideal-closure operators on ω-CPOs. The main technical result of our paper is a translation of † in terms of * that gives an embedding of the (in)equational theory of † in a typed variant of KA with products. Informally, this says that we can reason about the basic equational properties of fixpoints using the language of regular expressions and the axioms of Kleene algebra.

This work has been a first step in presenting a higher-order system of typed Kleene algebra. We would like to investigate what properties of recursion can be captured in such a higher-order system and how this would relate to the investigations of [19].

CHAPTER 5

CONCLUSION

The general theme of this dissertation has been the study of variations and extensions of KA and KAT that are useful for the verification of computer programs. KA and KAT enable equational reasoning for programs, which subsumes other approaches in program verification based on Hoare logic. One of the features that makes these systems so versatile and useful for verification is that they can accommodate in a straightforward way extra assumptions, thus capturing crucial properties of the domain of computation.

We believe that the previous successes of KA and KAT and the results presented here provide sufficient evidence to support the claim that formal systems based on KAT can play an increasingly important rôle in the field of program verification. A notable example towards this direction has been NetKAT [6], a KAT-based system that enables equational reasoning for software-define networks (SDNs). There are many open questions and opportunities for future research regarding how to accommodate parallelism and concurrency in KA [54], how to reason about probabilistic programs equationally [63, 64, 66, 67], and how to cover many more features of computation.

The connection between language models of (variants of) KA that characterize the equational theory and automata-theoretic models has been explored in a generalized setting using the framework of *coalgebra* [116, 117, 119]. Such operational characterizations in terms of machine models give rise to decision procedures based on the notion of bisimulation. We believe that these techniques will continue to inspire practical decision procedures for variations of KA.

BIBLIOGRAPHY

[1] Samson Abramsky and Achim Jung. Domain theory. In *Handbook of Logic in Computer Science*, volume 3, pages 1–168. 1994.

[2] Luca Aceto, Wan Fokkink, and Anna Ingólfsdóttir. On a question of A. Salomaa – The equational theory of regular expressions over a singleton alphabet is not finitely based. *Theoretical Computer Science*, 209(1):163–178, 1998.

[3] Jirí Adámek, Stefan Milius, and Jirí Velebil. Elgot algebras. *Logical Methods in Computer Science*, 2(5):1–31, 2006.

[4] Jirí Adámek, Stefan Milius, and Jirí Velebil. Elgot theories: A new perspective of the equational properties of iteration. *Mathematical Structures in Computer Science*, 21(2):417–480, 2011.

[5] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[6] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. NetKAT: Semantic foundations for networks. In *Proceedings of the 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '14)*, pages 113–126, 2014.

[7] Allegra Angus and Dexter Kozen. Kleene algebra with tests and program schematology. Technical Report TR2001-1844, CS Department, Cornell University, July 2001.

[8] Krzysztof R. Apt. Ten years of Hoare's logic: A survey – Part I. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 3(4):431–483, 1981.

[9] Krzysztof R. Apt. Ten years of Hoare's logic: A survey – Part II: Nondeterminism. *Theoretical Computer Science*, 28(1):83–109, 1983.

[10] Edward Ashcroft and Zohar Manna. The translation of GOTO programs into WHILE programs. In *Proceedings of IFIP Congress '71*, volume 1, pages 250–255. North-Holland, 1972.

[11] Edward Ashcroft, Zohar Manna, and Amir Pnueli. Decidable properties of monadic functional schemas. *Journal of the ACM*, 20(3):489–499, 1973.

[12] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[13] Philippe Balbiani, Andreas Herzig, and Nicolas Troquard. Dynamic logic of propositional assignments: A well-behaved variant of PDL. In *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '13)*, pages 143–152, 2013.

[14] Adam Barth and Dexter Kozen. Equational verification of cache blocking in LU decomposition using Kleene algebra with tests. Technical Report TR2002-1865, Computer Science Department, Cornell University, June 2002.

[15] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[16] Stephen L. Bloom and Zoltán Ésik. Equational axioms for regular sets. *Mathematical Structures in Computer Science*, 3(1):1–24, 1993.

[17] Stephen L. Bloom and Zoltán Ésik. *Iteration Theories: The Equational Logic of Iterative Processes*. Springer-Verlag New York, Inc., 1993.

[18] Stephen L. Bloom and Zoltán Ésik. Matrix and matricial iteration theories, part I. *Journal of Computer and System Sciences*, 46(3):381–408, 1993.

[19] Stephen L. Bloom and Zoltán Ésik. Fixed-point operations on ccc's. Part I. *Theoretical Computer Science*, 155(1):1–38, 1996.

[20] Stephen L. Bloom and Zoltán Ésik. The equational logic of fixed points. *Theoretical Computer Science*, 179(1–2):1–60, 1997.

[21] Corrado Böhm and Giuseppe Jacopini. Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM*, 9(5):366–371, 1966.

[22] Filippo Bonchi and Damien Pous. Checking NFA equivalence with bisimulations up to congruence. In *Proceedings of the 40th Annual ACM*

SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '13), pages 457–468, 2013.

[23] Ronald V. Book and Friedrich Otto. *String-Rewriting Systems*. Springer-Verlag, 1993.

[24] Thomas Braibant and Damien Pous. An efficient Coq tactic for deciding Kleene algebras. In Matt Kaufmann and Lawrence C. Paulson, editors, *Proceedings of the 1st International Conference on Interactive Theorem Proving (ITP '10)*, volume 6172 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 2010.

[25] Thomas Braibant and Damien Pous. Deciding Kleene algebras in Coq. *Logical Methods in Computer Science*, 8(1):16:1–42, 2012.

[26] Ashok K. Chandra. *On the Properties and Applications of Program Schemas*. PhD thesis, Stanford University, 1973.

[27] Ernie Cohen. Hypotheses in Kleene algebra. Technical report, Bellcore, 1993.

[28] Ernie Cohen. Lazy caching in Kleene algebra, 1994.

[29] Ernie Cohen. Using Kleene algebra to reason about concurrency control. Technical report, Telcordia, Morristown, N.J., 1994.

[30] Robert L. Constable and David Gries. On classes of program schemata. *SIAM Journal on Computing*, 1(1):66–118, 1972.

[31] John Horton Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, London, 1971.

[32] Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM Journal on Computing*, 7(1):70–90, 1978.

[33] D. C. Cooper. Program scheme equivalences and second-order logic. In *Machine Intelligence 4*, pages 3–15. Edinburgh University Press, 1969.

[34] Zoltán Ésik. Completeness of Park induction. *Theoretical Computer Science*, 177(1):217–283, 1997.

[35] Michael J. Fischer and Richard E. Ladner. Propositional modal logic of programs. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing (STOC '77)*, pages 286–294, 1977.

[36] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[37] John G. Fletcher. A more general algorithm for computing closed semiring costs between vertices of a directed graph. *Communications of the ACM*, 23(6):350–351, 1980.

[38] Robert W. Floyd. Assigning meanings to programs. In *Mathematical Aspects of Computer Science, Proceedings of AMS Symposium in Applied Mathematics*, volume 19, pages 19–32, 1967.

[39] Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. A coalgebraic decision procedure for NetKAT. In *Proceedings of the 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '15)*, pages 343–355, Mumbai, India, January 2015.

[40] Stephen J. Garland and David C. Luckham. On the equivalence of schemes. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing (STOC '72)*, pages 65–72, 1972.

[41] Stephen J. Garland and David C. Luckham. Program schemes, recursion schemes, and formal languages. *Journal of Computer and System Sciences*, 7(2):119–160, 1973.

[42] Susanna Ginali. Regular trees and the free iterative theory. *Journal of Computer and System Sciences*, 18(3):228–242, 1979.

[43] Sergey Goncharov. *Kleene Monads*. PhD thesis, Universität Bremen, 2010.

[44] Niels Bjørn Bugge Grathwohl, Dexter Kozen, and Konstantinos Mamouras. KAT + B! In *Proceedings of the Joint Meeting of the 23rd EACSL Conference on Computer Science Logic and the 29th ACM/IEEE Symposium on Logic in Computer Science (CSL-LICS '14)*, Vienna, Austria, July 2014.

[45] Chris Hardin and Dexter Kozen. On the elimination of hypotheses in Kleene algebra with tests. Technical Report TR2002-1879, CS Department, Cornell University, 2002.

[46] David Harel. On folk theorems. *Communications of the ACM*, 23(7):379–389, 1980.

[47] David Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 165 of *Synthese Library*, pages 497–604. Springer, 1984.

[48] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[49] David Harel, Dexter Kozen, and Jerzy Tiuryn. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 4 of *Handbook of Philosophical Logic*, pages 99–217. Springer, 2002.

[50] David Harel, Albert R. Meyer, and Vaughan R. Pratt. Computability and completeness in logics of programs (preliminary report). In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing (STOC '77)*, pages 261–268, 1977.

[51] David Harel and Vaughan R. Pratt. Nondeterminism in logics of programs. In *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '78)*, pages 203–213, 1978.

[52] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580,583, 1969.

[53] C. A. R. Hoare. Procedures and parameters: An axiomatic approach. In *Symposium on Semantics of Algorithmic Languages*, pages 102–116. Springer, 1971.

[54] Tony Hoare, Bernhard Möller, Georg Struth, and Ian Wehrman. Concurrent Kleene algebra and its foundations. *The Journal of Logic and Algebraic Programming*, 80(6):266–296, 2011.

[55] Yu I. Ianov. The logical schemes of algorithms. *Problems of Cybernetics*, 1:82–140, 1960.

[56] Kazuo Iwano and Kenneth Steiglitz. A semiring on convex polygons and zero-sum cycle problems. *SIAM Journal on Computing*, 19(5):883–901, 1990.

[57] Donald M. Kaplan. Regular expressions and the equivalence of programs. *Journal of Computer and System Sciences*, 3(4):361–386, 1969.

[58] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. Technical Report RM-704, RAND Corporation, 1951.

[59] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. In Claude E. Shannon and John McCarthy, editors, *Automata Studies*, number 34 in Annals of Mathematics Studies, pages 3–41. Princeton University Press, 1956.

[60] S. Rao Kosaraju. Analysis of structured programs. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing (STOC '73)*, pages 240–252, 1973.

[61] S. Rao Kosaraju. Analysis of structured programs. *Journal of Computer and System Sciences*, 9(3):232–255, 1974.

[62] Łucja Kot and Dexter Kozen. Kleene algebra and bytecode verification. *ENTCS*, 141(1):221–236, 2005.

[63] Dexter Kozen. Semantics of probabilistic programs. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS '79)*, pages 101–114. IEEE, 1979.

[64] Dexter Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981.

[65] Dexter Kozen. On induction vs. *-continuity. In Dexter Kozen, editor, *Proceedings of Workshop on Logics of Programs (1981)*, volume 131 of *Lecture Notes in Computer Science*, pages 167–176. Springer, 1982.

[66] Dexter Kozen. A probabilistic PDL. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC '83)*, pages 291–297, 1983.

[67] Dexter Kozen. A probabilistic PDL. *Journal of Computer and System Sciences*, 30(2):162–178, 1985.

[68] Dexter Kozen. On Kleene algebras and closed semirings. In *Proceedings of the 15th Symposium on Mathematical Foundations of Computer Science (MFCS '90)*, pages 26–47, 1990.

[69] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proceedings of Sixth Annual IEEE Symposium on Logic in Computer Science (LICS '91)*, pages 214–225, 1991.

[70] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[71] Dexter Kozen. Kleene algebra with tests and commutativity conditions. In *Proceedings of the 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '96)*, volume 1055 of *Lecture Notes in Computer Science*, pages 14–33. Springer, 1996.

[72] Dexter Kozen. Kleene algebra with tests. *Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997.

[73] Dexter Kozen. On the complexity of reasoning in Kleene algebra. In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science (LICS '97)*, pages 195–202, 1997.

[74] Dexter Kozen. Typed Kleene algebra. Technical report, Cornell University, 1998.

[75] Dexter Kozen. On Hoare logic and Kleene algebra with tests. *ACM Transactions on Computational Logic*, 1(1):60–76, 2000.

[76] Dexter Kozen. Halting and equivalence of schemes over recursive theories. Technical Report TR2002-1881, Computer Science Department, Cornell University, October 2002.

[77] Dexter Kozen. On the complexity of reasoning in Kleene algebra. *Information and Computation*, 179:152–162, 2002.

[78] Dexter Kozen. Some results in dynamic model theory. *Science of Computer Programming*, 51(1-2):3–22, 2004.

[79] Dexter Kozen and Konstantinos Mamouras. Kleene algebra with products and iteration theories. In *Proceedings of the 22nd EACSL Annual Conference on Computer Science Logic (CSL '13)*, pages 415–431, 2013.

[80] Dexter Kozen and Konstantinos Mamouras. Kleene algebra with equations. In *Proceedings of the 41st International Colloquium on Automata, Lan-*

guages and Programming (ICALP '14), Part II, volume 8573 of *Lecture Notes in Computer Science*, pages 280–292. Springer, 2014.

[81] Dexter Kozen and Maria-Cristina Patron. Certification of compiler optimizations using Kleene algebra with tests. In *Proceedings of the First International Conference on Computational Logic (CL '00)*, pages 568–582, 2000.

[82] Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In *Proceedings of the 10th International Workshop on Computer Science Logic (CSL'96)*, pages 244–259. Springer-Verlag, 1996.

[83] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 789–840. North Holland, 1990.

[84] Dexter Kozen and Jerzy Tiuryn. On the completeness of propositional Hoare logic. *Information Sciences*, 139(3-4):187–195, 2001.

[85] Dexter Kozen and Wei-Lung (Dustin) Tseng. The Böhm-Jacopini theorem is false, propositionally. In *Proceedings of the 9th International Conference on Mathematics of Program Construction (MPC '08)*, pages 177–192, 2008.

[86] Dexter C. Kozen. *The Design and Analysis of Algorithms*. Springer-Verlag, New York, 1991.

[87] Daniel Krob. A complete system of B-rational identities. In *Proceedings of 17th International Colloquium on the Automata, Languages and Programming (ICALP '90)*, pages 60–73. 1990.

[88] Daniel Krob. Complete systems of B-rational identities. *Theoretical Computer Science*, 89(2):207–343, 1991.

[89] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*, volume 5 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1986.

[90] David C. Luckham, David M. R. Park, and Michael S. Paterson. On formalised computer programs. *Journal of Computer and System Sciences*, 4(3):220–249, 1970.

[91] Konstantinos Mamouras. On the Hoare theory of monadic recursion schemes. In *Proceedings of the Joint Meeting of the 23rd EACSL Annual Con-*

ference on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, pages 69:1–69:10, 2014.

[92] Konstantinos Mamouras. Synthesis of strategies and the Hoare logic of angelic nondeterminism. In Andrew Pitts, editor, *Proceedings of the 18th International Conference on Foundations of Software Science and Computation Structures (FOSSACS '15)*, volume 9034 of *Lecture Notes in Computer Science*, pages 25–40. Springer, 2015.

[93] Zohar Manna. *Termination of Algorithms*. PhD thesis, Carnegie-Mellon University, 1968.

[94] Zohar Manna. *Mathematical Theory of Computation*. McGraw-Hill, 1974.

[95] Kurt Mehlhorn. *Data Structures and Algorithms 2: Graph Algorithms and NP-Completeness*, volume 2 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1984.

[96] Grażyna Mirkowska. *Algorithmic Logic and its Applications in the Theory of Programs*. PhD thesis, 1972.

[97] Kan Ching Ng. *Relation Algebras with Transitive Closures*. PhD thesis, University of California, Berkeley, 1984.

[98] Gordon Oulsnam. Unravelling unstructured programs. *The Computer Journal*, 25(3):379–387, 1982.

[99] Michael S. Paterson. *Equivalence Problems in a Model of Computation*. PhD thesis, University of Cambridge, 1967.

[100] Michael S. Paterson. Program schemata. In *Machine Intelligence 3*, pages 19–31. Edinburgh University Press, 1968.

[101] Michael S. Paterson. Decision problems in computational models. In *Proceedings of ACM Conference on Proving Assertions About Programs*, pages 74–82, 1972.

[102] Michael S. Paterson and Carl E. Hewitt. Comparative schematology. In Jack B. Dennis, editor, *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, pages 119–127. ACM, 1970.

[103] W. Wesley Peterson, Tadao Kasami, and Nobuki Tokura. On the capabilities of while, repeat, and exit statements. *Communications of the ACM*, 16(8):503–512, 1973.

[104] Gordon Plotkin. Domains, 1983. Pisa Notes on Domain Theory.

[105] Gordon Plotkin and John Power. Computational effects and operations. *ENTCS*, 73:149–163, 2004.

[106] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings of the 17th IEEE Annual Symposium on Foundations of Computer Science (FOCS '76)*, pages 109–121, 1976.

[107] Vaughan R. Pratt. Models of program logics. In *Proceedings of the 20th IEEE Annual Symposium on Foundations of Computer Science (FOCS '79)*, pages 115–122, 1979.

[108] Vaughan R. Pratt. Dynamic algebras and the nature of induction. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC '80)*, pages 22–28, 1980.

[109] Vaughan R. Pratt. Dynamic algebras as a well-behaved fragment of relation algebras. In Clifford H. Bergman, Roger D. Maddux, and Don L. Pigozzi, editors, *Proceedings of Conference on Algebraic Logic and Universal Algebra in Computer Science (1988)*, volume 425 of *Lecture Notes in Computer Science*, pages 77–110. Springer, 1990.

[110] Joseph E. Qualitz. *Equivalence Problems for Monadic Schemas*. PhD thesis, Massachusetts Institute of Technology, 1975.

[111] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[112] Lyle Ramshaw. Eliminating go to's while preserving program structure. *Journal of the ACM*, 35(4):893–920, 1988.

[113] Valentin N. Redko. On defining relations for the algebra of regular events. *Ukrainskii Matematicheskii Zhurnal*, 16:120–126, 1964.

[114] Jurriaan Rot, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Coalgebraic bisimulation-up-to. In *Proceedings of SOFSEM '13: Theory and Practice of Computer Science*, pages 369–381, 2013.

[115] Joseph D. Rutledge. On Ianov's program schemata. *Journal of the ACM*, 11(1):1–9, 1964.

[116] Jan J. M. M. Rutten. Automata and coinduction (an exercise in coalgebra). In Davide Sangiorgi and Robert de Simone, editors, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR '98)*, volume 1466 of *Lecture Notes in Computer Science*, pages 194–218. Springer, 1998.

[117] Jan J. M. M. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[118] Arto Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the ACM*, 13(1):158–169, 1966.

[119] Alexandra Silva. *Kleene Coalgebra*. PhD thesis, Centrum Wiskunde & Informatica, 2010.

[120] Alex Simpson and Gordon Plotkin. Complete axioms for categorical fixed-point operators. In *Proceedings of 15th Annual IEEE Symposium on Logic in Computer Science (LICS '00)*, pages 30–41, 2000.

[121] H. R. Strong. High level languages of maximum power. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (SWAT '71)*, pages 1–4, 1971.

[122] H. R. Strong, Jr. Translating recursion equations into flow charts. In *Proceedings of the Second Annual ACM Symposium on Theory of Computing (STOC '70)*, pages 184–197, 1970.

[123] H. R. Strong, Jr. Translating recursion equations into flow charts. *Journal of Computer and System Sciences*, 5(3):254–285, 1971.

[124] Alfred Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.

[125] M. Howard Williams and H. L. Ossher. Conversion of unstructured flow diagrams to structured form. *The Computer Journal*, 21(2):161–167, 1978.