1
Introduction

This book is about topics in category theory motivated by theoretical computer science. The study of initial algebras and terminal coalgebras is a lively area rather than a well-established field. We present a topical picture but do not aim for the last word on the subject. Our goal in this opening chapter is to motivate the book and its overall field for readers who open it without understanding even the title. What are initial algebras, terminal coalgebras, and fixed points of functors? Why do the authors care about them? What do we hope you will learn? What background will you need?

1.1 Why Are Initial Algebras and Terminal Coalgebras Interesting?

Recursion and induction are important tools in mathematics and computer science. Recursion is a definition principle for functions over (inductive) structures of data types such as natural numbers, lists, or trees. Induction is the corresponding proof principle used to prove properties of functions defined by recursion. An important question of theoretical computer science concerns the semantics of such definitions. *Initial Algebra Semantics*, studied since the 1970s, uses the tools of category theory to unify recursion and induction at the appropriate abstract conceptual level. In this approach, the type of data on which one wants to define functions recursively and to prove properties inductively is captured by an *endofunctor* F on the category of sets (or another appropriate base category such as posets, metric spaces, or vector spaces). This functor describes the signature of the data type constructors. An *F-algebra* is an object A together with a morphism $\alpha\colon FA \to A$. An initial algebra for the functor F provides a canonical model of a data type with the desired constructors. Typically the initial algebra is the intended model, and then one tries to use initiality

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

to prove properties of various definitions, such as equivalence of programs or specifications.

Let us illustrate this with the simplest concrete example: Consider the endofunctor on sets given by $FX = X + 1$. This is the construction adding a fresh element to the set X, made into a functor in the obvious way. An algebra for F is just a set A equipped with a unary operation $u\colon X \to X$ and an element $a \in A$. We combine u and a into a function $\alpha\colon A + 1 \to A$. We are going to properly define algebras and their morphisms below along with initial algebras. For now, we simply assert that the initial algebra of our functor F is the algebra of natural numbers $\mathbb{N}$ with the successor function $s(n) = n + 1$ and the constant 0. The initiality of this algebra is equivalent to the usual principle of *recursion on natural numbers*: given an F-algebra (A, α), there exists precisely one function $h\colon \mathbb{N} \to A$ with $h(0) = a$ and $h(s(n)) = u(h(n))$ for all natural numbers n. In other words, functions from the initial algebra to another algebra can be defined by recursion.

As a second example, consider the set functor $FX = X \times X + 1$. An algebra $\alpha\colon A \times A + 1 \to A$ consists of a set equipped with a binary operation and a constant. This time, the initial algebra is the algebra of finite binary trees, and initiality yields a tree-recursion principle which is quite parallel to the induction principle for $\mathbb{N}$.

A collection of objects 'built from below' (such as the natural numbers or the finite trees) is typically described as an initial algebra of some set functor. Addressing our opening question: initial algebras are interesting because they are the natural objects with which to capture recursion and induction.

A *coalgebra* for a functor F is the dual concept of an F-algebra: it consists of an object A and a morphism $\alpha\colon A \to FA$. We think of A as the set of states in a system, and α assigning to a given state a structured collection formed by its successor states modelling the one-step behaviour of the state.

For example, a deterministic automaton with input alphabet Σ can be described by the set A of its states together with a function

$$\alpha\colon A \to \{0, 1\} \times A^{\Sigma}$$

whose first component $\mathsf{acc}\colon A \to \{0, 1\}$ describes the yes/no predicate of being an accepting state. The second component $A \to A^{\Sigma}$ may be canonically recast as $\mathsf{next}\colon A \times \Sigma \to A$. This is the next-state function of the automaton. Thus, a deterministic automaton (with no initial state specified) is a coalgebra for the set functor F given by

$$FX = \{0, 1\} \times X^{\Sigma}.$$

As with initial algebras, we mention an example of a terminal coalgebra before

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

giving the definitions. In the case of deterministic automata, let Σ^* be the set of words on the alphabet Σ, and let $\mathcal{L} = \mathcal{P}(\Sigma^*)$ be the set of (formal) languages over Σ. Then $\mathcal{L}$ itself has a natural automaton structure: the next-state function assigns to a language L and an input symbol σ the language $\partial_\sigma L$ of all words w with $\sigma w \in L$. The accepting states are the languages containing the empty word. This automaton turns out to be the terminal coalgebra. To explain this, we must mention the concept of a *coalgebra homomorphism*. In general, this is a morphism in the base category that respect the coalgebra structure. In the special case of deterministic automata, this would be a function between the state sets that respects the accepting-state predicate and the next-state function. Such functions are called *strong simulations* in automata theory [96]. For a given automaton (A, α), we consider the map $\mathsf{lang}\colon A \to \mathcal{L}$ assigning to a state a the language L_a of words accepted by A when started in a. Then $\mathsf{acc}(a) = 1$ iff $\varepsilon \in L_a$ iff L is an accepting state of $\mathcal{L}$. Furthermore, $\partial_\sigma(L_a) = L_{\mathsf{next}(a,\sigma)}$. This verifies that lang is a coalgebra homomorphism. With further work, one verifies that lang is the unique such function. This is what it means to say that $\mathcal{L}$ is a *terminal coalgebra*.

For another example, take an input alphabet Σ and an output alphabet Γ. A transducer is an automaton which reads in infinite words on Σ and outputs infinite words on Γ according to a next-state function, which may be described as a function of type $A \times \Sigma \to A \times \Gamma$. Closer to our purposes, we transpose it into a function $\alpha\colon A \to (A \times \Gamma)^\Sigma$. Thus, we are considering the set functor $FX = (X \times \Gamma)^\Sigma$. Transducers are precisely coalgebras for F. Let Σ^ω and Δ^ω be the set of infinite sequences of elements of Σ and Δ, respectively. Let C be the set of functions $f\colon \Sigma^\omega \to \Delta^\omega$ which are *causal*: the nth entry in $f(\sigma_1\sigma_2\cdots)$ depends only on $\sigma_1\sigma_2\cdots\sigma_n$. The terminal coalgebra is carried by this set C. We leave it to the reader to guess the terminal coalgebra structure $C \to (C \times \Gamma)^\Sigma$.

The point is that various examples of state-based systems of various sorts may be taken to be coalgebras for set functors. Then the terminal coalgebra appears as the 'intended semantic space'. Terminal coalgebras for set functors have an infinitary aspect that contrasts with what we saw in the most basic initial algebras. For example, languages $L \in \mathcal{L}$ usually are infinite sets of words, and the elements of Σ^ω and Δ^ω are infinite sequences. Dually to recursion and induction, there are principles of function definition by *corecursion* and proof by *coinduction*. Proofs by coinduction are a bit more difficult to illustrate; we present examples in Section 2.6. For now, let us circle back to the title of this section. Terminal coalgebras are interesting because the intended semantic spaces associated with various forms of automata appear as terminal coalgebras. In addition, they are tied up with corecursion, an interesting dual to recursion.

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

So to the extent that we are interested in recursion (hence in initial algebras), we should also be at least curious about its cousin, corecursion.

We turn next to the formal definitions of algebra and coalgebra, presented for all categories $\mathscr{A}$ and endofunctors $F\colon \mathscr{A} \to \mathscr{A}$.[1] (Incidentally, when $\mathscr{A}$ is the category Set of all sets and functions, we speak of *set functors* in lieu of endofunctors on Set.) An *F-algebra* is a pair (A, α) consisting of an object A of $\mathscr{A}$ and a morphism $\alpha\colon FA \to A$; the *F-algebra homomorphisms* (shortly *homomorphisms*) from (A, α) to (B, β) are those morphisms $h\colon A \to B$ of $\mathscr{A}$ for which the square below commutes:

$$\begin{array}{ccc} FA & \xrightarrow{\alpha} & A \\ {\scriptstyle Fh}\downarrow & & \downarrow{\scriptstyle h} \\ FB & \xrightarrow{\beta} & B \end{array}$$

The category of algebras is denoted by $\mathsf{Alg}\, F$. By an *initial algebra* μF of F is meant the initial object in $\mathsf{Alg}\, F$ (if it exists): this is an algebra such that for every algebra, there exists a unique homomorphism from μF.

When we mentioned $\mathbb{N}$ as an initial algebra for the set functor $FX = X + 1$, we stated the initiality property in concrete terms. Now we wish to view the initiality of $\mathbb{N}$ as a special case of the general definitions which we have just seen. Suppose we are given as set A and a function $\alpha\colon A + 1 \to A$. From α we can extract a function $u\colon A \to A$ and an element $a\colon 1 \to A$. Then the initiality of $\mathbb{N}$ means that there is a unique function $h\colon \mathbb{N} \to A$ as follows:

$$\begin{array}{ccc} \mathbb{N} + 1 & \xrightarrow{[s,0]} & \mathbb{N} \\ {\scriptstyle h+\mathsf{id}}\downarrow & & \downarrow{\scriptstyle h} \\ A + 1 & \xrightarrow{\alpha=[u,a]} & A \end{array}$$

How is initiality connected to definition by recursion? Following the element of the singleton set 1 around the square shows that $h(0) = a$, and following a number n in $\mathbb{N}$ shows that $h(s(n)) = u(h(n))$. So h satisfies the recursion equation driven by u and a. Conversely, any function h' which happens to satisfy those recursion equations is an algebra morphism. So by the uniqueness part of initiality, h and h' must be the same. This translates back to the fact that functions which satisfy recursion equations do so uniquely. The overall point is that recursion on the natural numbers is tantamount to initiality of $(\mathbb{N}, [s, 0])$ in $\mathsf{Alg}(F)$, where $FX = X + 1$.

For the coalgebraic concepts, we turn the structure morphisms around. An *F-coalgebra* is a pair (A, α) with $\alpha\colon A \to FA$. An *F-coalgebra homomorphism*

[1] This book properly begins in Chapter 2. What we are doing here is presenting just a few definitions so that we can discuss the content of the book in a very general way.

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

(shortly: *homomorphism*) from (A, α) to (B, β) is a morphism h of $\mathscr{A}$ such that $Fh \cdot \alpha = \beta \cdot h$:

$$\begin{array}{ccc} A & \xrightarrow{\alpha} & FA \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle Fh} \\ B & \xrightarrow{\beta} & FB \end{array}$$

This defines a category $\mathsf{Coalg}\, F$ of coalgebras. The *terminal coalgebra* νF is (if it exists) the terminal object of this category.

We have already seen a homomorphism when we discussed $\mathsf{lang}\colon A \to \mathcal{L}$ in connection with deterministic automata. The unique homomorphism from an automaton into the terminal coalgebra assigns to every state its language. We see again an instance of a general trend: the unique homomorphism from a coalgebra to νF assigns to every state its 'behaviour'. This underscores the importance of terminal coalgebras. We make a junction between computer science and category theory when we see that concepts like language acceptance satisfy universal properties every bit as compelling, beautiful, and useful as those in other areas of mathematics. This is one of the reasons to be interested in the coalgebraic modelling of dynamical systems of all sorts: it leads to a generic description of behaviour-preserving maps between those systems and an ensuing notion of behavioural equivalence of states. Two states are *behaviourally equivalent* if they are merged by some coalgebra homomorphism. This specializes to well-known notions of behavioural equivalence, for example language equivalence in the case of deterministic automata or Milner's (strong) bisimilarity for labelled transition systems (modelled as coalgebras). This observation was made forcefully by Jan Rutten [264]; he presented a persuasive survey of applications of coalgebra to the theory of discrete dynamical systems. Aczel and Mendler [4] made another decisive contribution by proposing a categorical formulation of bisimulation which revised our conception of everything that went before it in this area and attracted the interest of a new generation of computer scientists. One of the motivations for this book is to present a selection of the large body of work inspired by the papers of Rutten and Aczel–Mendler.

Let us return to homomorphisms, this time of algebras. These are the maps preserving operations, so they are well-known from abstract algebra. The interest of initial algebras for computer science is bolstered because they provide minimal realizations of abstract data type specifications as terms possibly modulo equations.

We shall see in Section 1.3 that initial algebras and terminal coalgebras do not always exists. When do they exist? How can we describe them when they exist? If one exists, does the other? These questions turn out to be difficult,

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

and there are no easily-quoted or complete answers. A large part of this book addresses them. The most basic construction is discussed in Chapter 3. It applies to both initial algebras and terminal coalgebras. This leads to a more in-depth look at certain terminal coalgebras in sets in Chapter 4, and to variations that work in complete partial orders and complete metric spaces in Chapter 5. These variations provide settings where μF and νF are the same. That is, we have an initial algebra (A, α) with the property that (A, α^{-1}) is a terminal coalgebra.

Other chapters which take up the theme of constructing and describing initial algebras and terminal coalgebras include Chapters 6, 11, 12, 13, 15, and Appendix A.

1.2 Expected Background

We intend the book to be our part of the deep connection of category theory to computer science. The root of the matter is that category theory is the mathematics of *structure* par excellence and the semantic side of computer science is full of issues that revolve around mathematical structure.

What background is needed to read this book? We are mainly writing to readers who already know about basic concepts in category theory: initial and terminal objects, duality, products and coproducts (and limits/colimits more generally), monomorphisms, and epimorphisms. Very little else is required. Later chapters in the book require more, of course. The topics that would be most helpful for later work include adjunctions, monads, factorization systems, (co)complete categories, and locally finitely presentable categories.

Even more than knowledge of the specific topics, what we assume is a level of comfort with the presentational style that comes from category theory. Let us illustrate this point by presenting a very simple result having to do with an abstract formulation of induction. The rest of this section contains that result and, in addition, it is a comment on it and the tone and reasoning style of this book.

We mentioned at the outset that initiality is connected with recursion and induction, but we have in fact not said a word about the connection to induction. Now that we have the category $\mathsf{Alg}(F)$, we can connect initiality and induction. Let $\mathscr{A}$ be a category, and let $F\colon \mathscr{A} \to \mathscr{A}$. If (A, α) is an algebra, a *subalgebra* of it is represented by a homomorphism $m\colon (B, \beta) \rightarrowtail (A, \alpha)$ which is a monomorphism in $\mathscr{A}$ (hence the notation $\rightarrowtail$). An algebra (A, α) is *minimal* if every $m\colon (B, \beta) \rightarrowtail (A, \alpha)$ is an isomorphism.

We recall from category theory that a *split epimorphism* is a morphism $e\colon X \to Y$ for which $m\colon Y \to X$ exists with $e \cdot m = \mathrm{id}_Y$. The reader will

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

recall that if a morphism is both a split epimorphism and a monomorphism, then it is an isomorphism.

Proposition 1.2.1. *Every initial algebra* $(\mu F, \iota)$ *is minimal.*

Proof By initiality, there is an algebra homomorphism $f\colon (\mu F, \iota) \to (B, \beta)$. The composition $m \cdot f$ is a homomorphism from $(\mu F, \iota)$ to itself, so by initiality again, $m \cdot f = \mathrm{id}_{\mu F}$. Hence m is a split epimorphism and a monomorphism, so it is an isomorphism. □

This general result specializes to well-known cases. For the set functor $FX = X + 1$, it corresponds to *weak induction*: every subset of $\mathbb{N}$ which contains 0 and is closed under successors contains all natural numbers.

There are other ways to formulate induction. For a parallel pair $f_1, f_2\colon \mu F \to A$ of morphisms in the base category $\mathscr{A}$ with domain μF, in order to prove $f_1 = f_2$ it is sufficient to present a morphism $\alpha\colon FA \to A$ for which f_1 and f_2 are algebra homomorphisms from $(\mu F, \iota)$ to (A, α). Notice that this is not the same formulation that we saw in Proposition 1.2.1. It is very much in the spirit of this book to investigate different categorial formulations of concepts like weak induction, strong induction, etc. *Coinduction* is the dual principle which for the terminal coalgebra νF allows us to prove equality of morphisms of the form $f_1, f_2\colon A \to \nu F$.

We return to the general point about the tone and reasoning style of the book. In introducing Proposition 1.2.1, we mentioned what a split epimorphism is and also one fact about it. Our brief mention is barely enough to read further. In other words, we want to jog the memories of people who have seen the definition, and we trust that others will acquire some background on it. The proof of Proposition 1.2.1 is shorter than almost any result in the book, but the reasoning style is indicative: it uses point-free category-theoretic arguments and makes use of universal properties.

We generally assume that readers know about automata, terms and signatures, equational logic, and other topics used in theoretical computer science. We do not assume much about any of those topics, but we take for granted that the reader is interested in them.

Several chapters require background on topics related to metric spaces or partially ordered sets. We have more background on that material. For example, we provide an appendix on fixed points in settings that come from complete metric spaces or complete partially ordered sets. We would like readers to come away from the book with appreciation of two things: the general theory, and results about particular concrete categories.

Let us comment on the aspect of the book connected to set theory. Our set

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

theory is the standard one using the Zermelo–Fraenkel axioms, including the Axiom of Choice. At times we do need definitions and results from basic set theory, touching on matters such as ordinal numbers, cardinal numbers, cardinal arithmetic, and regular infinite cardinals. We have provided a more thorough introduction to those topics in Chapter 6. We also collect a number of facts concerning set functors into a single place, Appendix C. Material from this appendix is used in most of the chapters. In a few other places, we mention some other concept in set theory (e.g. filters and ultrafilters), and readers who do not know about such things may skip those passages.

When speaking about a category $\mathscr{C}$ we assume that the collection $\mathscr{C}(X, Y)$ of all morphisms from X to Y is a set (not a proper class) for every pair of objects X, Y. We also assume that these hom-sets are pairwise disjoint.

1.3 Fixed Points and Special Kinds of Algebras and Coalgebras

As stated in its subtitle, our book is about the theory of fixed points of functors.

A *fixed point* of an endofunctor F is an object A together with an isomorphism of FA and A. One general reason to be interested in fixed points comes from the following basic result:

Lambek's Lemma. *Every initial algebra is a fixed point.*

Dually, every terminal coalgebra is a fixed point. Besides these two, there are other interesting fixed points, one of which we now mention.

In the study of state-based systems, one is mostly interested in *finite* systems (e.g. finite-state automata), or ones whose state space has a finite representation (e.g. a finite-dimensional vector space or an orbit-finite nominal set). For example, in classical automata theory the regular languages are those accepted by finite automata, and rational streams over a field are precisely those behaviours obtained from systems whose state space is a finite-dimensional vector space over the field.

We present a general treatment of regularity of system behaviour and the semantics of 'finite' coalgebras. For this we assume that the functor F is *finitary* (see Definition 4.3.1). For example, finitary set functors are those determined by their action on finite sets; for them the behaviour of all finite coalgebras yields a subcoalgebra of the terminal coalgebra νF (consisting of the behaviour of *all* coalgebras). This subcoalgebra turns out to be a fixed point of F which we call the *rational fixed point* and denote by ϱF. For example, for the functor $FX = \{0, 1\} \times X^{\Sigma}$ for deterministic automata, this is the coalgebra of all

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

regular languages over the alphabet Σ. For $FX = X \times X + 1$, the rational fixed point ϱF is the subcoalgebra of νF (of all binary trees) consisting of precisely those binary trees that have only finitely many subtrees (up to isomorphism); these are precisely Courcelle's *regular trees* [107]. In addition to the finite binary trees, the complete (infinite) binary tree is an example of a tree contained in ϱF.

The concept of a rational fixed point makes sense for endofunctors on categories other than Set, too. For example, for the functor F on the category of vector spaces whose coalgebras are linear weighted automata, we obtain the rational formal power series, known from weighted automata theory [113], as ϱF. We study rational fixed points in Chapter 10.

Coming back to initial algebras, it follows from Lambek's Lemma that an initial algebra can be considered as a coalgebra by taking the inverse of its structure morphism. It is interesting to study properties of this coalgebra, especially ones that characterize it in classes of coalgebras with computational relevance. We mention two properties here: *recursive* coalgebras and *well-founded* ones. Without going into the technical definitions, let us just say here that recursive coalgebras are coalgebras which admit function definitions by structural recursion; a good intuition here is to think about 'divide-and-conquer' algorithms. Well-founded coalgebras present the notion of a well-founded relation at a general coalgebraic level. So they are coalgebras admitting an induction principle. An initial algebra turns out to be the terminal recursive or well-founded coalgebra; the latter holds under mild assumptions. We shall study recursivity and well-foundedness in Chapters 7 and 8.

Yet another application of Lambek's Lemma, this time in dual form, lets us consider a terminal coalgebra νF as an algebra for F using the inverse structure. We shall be interested in studying properties which characterize νF in classes of algebras with computational relevance. *Completely iterative* algebras are connected to the matter of solving recursive equations. Again exemplifying via automata, we can solve systems of equations for languages in a unique way, such as $L = aM$, $M = bL + 1$. Completely iterative algebras are even tied up with areas of mathematics that at first blush have little to do with our subject: fractal sets, for example. A terminal coalgebra is, equivalently, an initial completely iterative algebra. Complete iterativity shall be studied in Chapter 7.

Finally, Lambek's Lemma also has a negative consequence for our study. It implies that even for $\mathscr{A} = \mathsf{Set}$, there are important endofunctors that do not have an initial algebra: the power-set functor $\mathscr{P}$, for example. A fundamental result of set theory known as Cantor's Theorem [99] states that no set A is in bijective correspondence with $\mathscr{P}A$. (The short proof may be found in Example 2.2.7(1).) So no set is a fixed point of $\mathscr{P}$.

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

algebra for a functor	coalgebra for a functor
initial algebra	terminal coalgebra
least fixed point	greatest fixed point
congruence relation	bisimulation equivalence relation
equational logic	modal logic
recursion: map out of an initial algebra	corecursion: map into a terminal coalgebra
iterative conception of set	coiterative conception of set
set with operations	set with transitions and observations
useful in syntax	useful in semantics
bottom-up	top-down

Figure 1.1 Conceptual comparison.

However, this negative point does raise questions about conditions on functors that ensure the existence of initial algebras and terminal coalgebras and results showing how to construct them. For example, what about functors related to $\mathscr{P}$, such as the finite power-set functor, or the countable power-set functor? Do these have fixed points? It is known that in other areas such as metric spaces or posets, fixed points do exist for important classes of functions. We review this general topic in Appendix B.

1.4 Algebraic versus Coalgebraic Concepts

Although *algebra* and *coalgebra* are dual terms, and although this duality persists to the level of *initial algebra* and *terminal coalgebra*, $\mathsf{Alg}\, F$ is not dual to $\mathsf{Coalg}\, F$. There are easy examples of this; here is a very simple one: for the constant functor with value 1, a one-element set, $\mathsf{Alg}\, F$ is the category of pointed sets, while $\mathsf{Coalg}\, F$ is (isomorphic to) the category of sets.

What is true is the following: every functor $F\colon \mathscr{A} \to \mathscr{A}$ defines a functor $F^{\mathsf{op}}\colon \mathscr{A}^{\mathsf{op}} \to \mathscr{A}^{\mathsf{op}}$ by the same rule as F. The category of algebras for F (in $\mathscr{A}$) is dual to the category of coalgebras for F^{op} (in $\mathscr{A}^{\mathsf{op}}$). Shortly,

$$(\mathsf{Alg}\, F)^{\mathsf{op}} = \mathsf{Coalg}(F^{\mathsf{op}}).$$

We present in Figure 1.1 a comparison between concepts and ideas in algebra and coalgebra. The algebraic concepts on the left are connected to *sets with operations* and the coalgebraic ones on the right are about sets with structure corresponding to *transitions* and *observations*. The two columns are duals, we are speaking of 'duals' here in an informal way. This book will not have much to say about most of this chart: to explore it in any depth would require several additional chapters at the least, and perhaps an entire book of its own. However,

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press

we hope that readers find this chart enticing and that they will want to explore the book in order to construct their own conceptual charts. Our claim in this book is that the mathematical tools which we study will be useful for anyone interested in the chart.

To sum up, this book is concerned with fundamental notions in theoretical computer science related to recursion and induction. These notions are dualized and generalized, leading to a wealth of questions and results. We bring the viewpoint of category theory to our subject, therefore we adopt the vocabulary and tool set from that theory. In addition to general work, we frequently study particular concrete categories and particular endofunctors.

https://doi.org/10.1017/9781108884112.002 Published online by Cambridge University Press