Advanced Topics in Bisimulation and Coinduction

Edited by Davide Sangiorgi
and Jan Rutten

CAMBRIDGE

CAMBRIDGE

more information - www.cambridge.org/9781107004979

Cambridge Tracts in Theoretical Computer Science 52

Advanced Topics in Bisimulation and Coinduction

Coinduction is a method for specifying and reasoning about infinite data types and automata with infinite behaviour. In recent years, it has come to play an ever more important role in the theory of computing. It is studied in many disciplines, including process theory and concurrency, modal logic and automata theory. Typically, coinductive proofs demonstrate the equivalence of two objects by constructing a suitable bisimulation relation between them.

This collection of surveys is aimed at both researchers and Master's students in computer science and mathematics, and deals with various aspects of bisimulation and coinduction, with an emphasis on process theory. Seven chapters cover the following topics: history; algebra and coalgebra; algorithmics; logic; higher-order languages; enhancements of the bisimulation proof method; and probabilities. Exercises are also included to help the reader master new material.

DAVIDE SANGIORGI is Full Professor in Computer Science at the University of Bologna, Italy, and Head of the University of Bologna/INRIA team 'Focus'.

JAN RUTTEN is a senior researcher at Centrum Wiskunde & Informatica (CWI) in Amsterdam and Professor of Theoretical Computer Science at the Radboud University Nijmegen.

Cambridge Tracts in Theoretical Computer Science 52

Editorial Board

Titles in the series

A complete list of books in the series can be found at
www.cambridge.org/mathematics.
Recent titles include the following:

29. P. Gärdenfors (ed) *Belief Revision*
30. M. Anthony & N. Biggs *Computational Learning Theory*
31. T. F. Melham *Higher Order Logic and Hardware Verification*
32. R. Carpenter *The Logic of Typed Feature Structures*
33. E. G. Manes *Predicate Transformer Semantics*
34. F. Nielson & H. R. Nielson *Two-Level Functional Languages*
35. L. M. G. Feijs & H. B. M. Jonkers *Formal Specification and Design*
36. S. Mauw & G. J. Veltink (eds) *Algebraic Specification of Communication Protocols*
37. V. Stavridou *Formal Methods in Circuit Design*
38. N. Shankar *Metamathematics, Machines and Gödel's Proof*
39. J. B. Paris *The Uncertain Reasoner's Companion*
40. J. Desel & J. Esparza *Free Choice Petri Nets*
41. J.-J. Ch. Meyer & W. van der Hoek *Epistemic Logic for AI and Computer Science*
42. J. R. Hindley *Basic Simple Type Theory*
43. A. S. Troelstra & H. Schwichtenberg *Basic Proof Theory*
44. J. Barwise & J. Seligman *Information Flow*
45. A. Asperti & S. Guerrini *The Optimal Implementation of Functional Programming Languages*
46. R. M. Amadio & P.-L. Curien *Domains and Lambda-Calculi*
47. W.-P. de Roever & K. Engelhardt *Data Refinement*
48. H. Kleine Büning & T. Lettmann *Propositional Logic*
49. L. Novak & A. Gibbons *Hybrid Graph Theory and Network Analysis*
50. J. C. M. Baeten, T. Basten & M. A. Reniers *Process Algebra: Equational Theories of Communicating Processes*
51. H. Simmons *Derivation and Computation*
52. D. Sangiorgi & J. Rutten (eds) *Advanced Topics in Bisimulation and Coinduction*
53. P. Blackburn, M. de Rijke & Y. Venema *Modal Logic*
54. W.-P. de Roever et al. *Concurrency Verification*
55. Terese *Term Rewriting Systems*
56. A. Bundy et al. *Rippling: Meta-Level Guidance for Mathematical Reasoning*

Advanced Topics in Bisimulation and Coinduction

Edited by

DAVIDE SANGIORGI
University of Bologna (Italy)
and INRIA (France)

JAN RUTTEN
Centrum Wiskunde & Informatica (CWI), Amsterdam
and the Radboud University Nijmegen

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Tokyo, Mexico City

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9781107004979

First published 2012

Printed in the United Kingdom at the University Press, Cambridge

A catalogue record for this publication is available from the British Library

ISBN 978-1-107-00497-9 Hardback

Additional resources for this publication at www.cs.unibo.it/∼sangio/Book_Bis_Coind.html

Contents

Contributors

Luca Aceto
School of Computer Science, Reykjavik University, Menntavegur 1,
101 Reykjavik, Iceland.
email: luca@ru.is
web: www.ru.is/faculty/luca/

Anna Ingolfsdottir
School of Computer Science, Reykjavik University, Menntavegur 1,
101 Reykjavik, Iceland.
email: annai@ru.is
web: www.ru.is/faculty/annai/

Bart Jacobs
Institute for Computing and Information Sciences (ICIS), Radboud University
Nijmegen, Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands.
email: bart@cs.ru.nl
web: www.cs.ru.nl/B.Jacobs/

Prakash Panangaden
McGill University, 3480 rue University, Room 318 Montreal, Quebec,
H3A 2A7 Canada.
email: prakash@cs.mcgill.ca
web: www.cs.mcgill.ca/∼prakash/

Andrew M. Pitts
University of Cambridge, Computer Laboratory, William Gates Building,
15 JJ Thomson Ave, Cambridge CB3 0FD, UK.
email: Andrew.Pitts@cl.cam.ac.uk
web: www.cl.cam.ac.uk/∼amp12/

Damien Pous
CNRS, Team Sardes, INRIA Rhône-Alpes, 655, avenue de l'Europe, Montbonnot, 38334 Saint Ismier, France.
email: Damien.Pous@inria.fr
web: http://sardes.inrialpes.fr/∼pous/

Jan Rutten
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands.
Also: Radboud University Nijmegen.
email: janr@cwi.nl
web: http://homepages.cwi.nl/∼janr/

Davide Sangiorgi
Università di Bologna/INRIA Team Focus, Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni, 7 40126 Bologna, Italy.
email: Davide.Sangiorgi@cs.unibo.it
web: www.cs.unibo.it/∼sangio/

Jiri Srba
Department of Computer Science, University of Aalborg, Selma Lagerlöfs Vej 300, 9220 Aalborg East, Denmark.
email: srba@cs.aau.dk
web: www.brics.dk/∼srba/

Colin Stirling
School of Informatics, Edinburgh University, Informatics Forum, 10 Crichton Street, Edinburgh EH8 9AB.
email: cps@staffmail.ed.ac.uk
web: http://homepages.inf.ed.ac.uk/cps/

Preface

This book is about bisimulation and coinduction. It is the companion book of the volume *An Introduction to Bisimulation and Coinduction*, by Davide Sangiorgi (Cambridge University Press, 2011), which deals with the basics of bisimulation and coinduction, with an emphasis on labelled transition systems, processes, and other notions from the theory of concurrency.

In the present volume, we have collected a number of chapters, by different authors, on several advanced topics in bisimulation and coinduction. These chapters either treat specific aspects of bisimulation and coinduction in great detail, including their history, algorithmics, enhanced proof methods and logic. Or they generalise the basic notions of bisimulation and coinduction to different or more general settings, such as coalgebra, higher-order languages and probabilistic systems. Below we briefly summarise the chapters in this volume.

- *The origins of bisimulation and coinduction, by Davide Sangiorgi*

  In this chapter, the origins of the notions of bisimulation and coinduction are traced back to different fields, notably computer science, modal logic, and set theory.
- *An introduction to (co)algebra and (co)induction, by Bart Jacobs and Jan Rutten*

  Here the notions of bisimulation and coinduction are explained in terms of coalgebras. These mathematical structures generalise all kinds of infinite-data structures and automata, including streams (infinite lists), deterministic and probabilistic automata, and labelled transition systems. Coalgebras are formally dual to algebras and it is this duality that is used to put both induction and coinduction into a common perspective. This generalises the treatment in the companion introductory volume, where induction and coinduction were explained in terms of least and greatest fixed points.

- *The algorithmics of bisimilarity, by Luca Aceto, Anna Ingolfsdottir and Jiří Srba*

  This chapter gives an overview of the solutions of various algorithmic problems relating bisimilarity and other equivalences and preorders on labelled transition systems. Typical questions that are addressed are: How can one compute bisimilarity? What is the complexity of the algorithms? When is bisimilarity decidable?
- *Bisimulation and logic, by Colin Stirling*

  This chapter discloses the strong and beautiful ties that relate bisimulation and modal logics. Various logical characterisations of bisimilarity are discussed. The main results are the characterisations of bisimilarity via a simple modal logic, the Hennessy–Milner logic, and the characterisation of this modal logic as the fragment of first-order logic that is bisimulation invariant. The results are then extended to modal logic with fixed points and to second-order logic.
- *Howe's Method for higher-order languages, by Andrew Pitts*

  In programming languages, an important property of bisimulation-based equivalences is whether they are a congruence, that is, compatible with the language constructs. This property may be difficult to prove if such languages involve higher-order constructs, that is, ones permitting functions and processes to be data that can be manipulated by functions and processes. This chapter presents a method for establishing compatibility of coinductively defined program equalities, originally due to Howe.
- *Enhancements of the bisimulation proof method, by Damien Pous and Davide Sangiorgi*

  This chapter discusses enhancements of the bisimulation proof method, with the goal of facilitating the proof of bisimilarity results. The bisimulation proof method is one of the main reasons for the success of bisimilarity. According to the method, to establish the bisimilarity between two given objects one has to find a bisimulation relation containing these objects as a pair. This means proving a certain closure property for each pair in the relation. The amount of work needed in proofs therefore depends on the size of the relation. The enhancements of the method in the chapter allow one to reduce such work by using relations that need only be *contained* in bisimulation relations. The chapter shows that it is possible to define a whole theory of enhancements, which can be very effective in applications.
- *Probabilistic bisimulation, by Prakash Panangaden*

  Here notions of bisimulation are introduced for probabilistic systems. These differ from non-deterministic ones in that they take quantitative data into account on the basis of which they make quantitative predictions about a

system's behaviour. The chapter first discusses the basic example of discrete systems, called labelled Markov chains. After a rapid introductory section on measure theory, the more general continuous case, of so-called labelled Markov processes, is treated. For both the discrete and the continuous case, logical characterisations of bisimilarity are given.

The chapters on probabilities and higher-order linguistic constructs deal with two important refinements of bisimulation. While these are certainly not the only interesting refinements of bisimulation (one could mention, for instance, the addition of time or of space constraints), probabilities and higher-order constructs have strong practical relevance (e.g. in distributed systems and other complex systems such as biological systems, and in programming languages) and offer technical challenges that make them one of the most active research topics in the area of coinduction and bisimulation.

Each chapter is a separate entity, therefore notations among chapters may occasionally differ. We are very grateful to the colleagues who contributed the chapters for the time, effort, and enthusiasm that they put into this book project.

We recall the Web page with general information and auxiliary material about the two volumes, including solutions to exercises in some of the chapters. At the time of writing, the page is

`www.cs.unibo.it/~sangio/Book_Bis_Coind.html`

Davide Sangiorgi and Jan Rutten

1

Origins of bisimulation and coinduction

DAVIDE SANGIORGI

1.1 Introduction

In this chapter, we look at the origins of bisimulation. We show that bisimulation has been discovered not only in computer science, but also – and roughly at the same time – in other fields: philosophical logic (more precisely, modal logic), and set theory. In each field, we discuss the main steps that led to the discovery, and introduce the people who made these steps possible.

In computer science, philosophical logic, and set theory, bisimulation has been derived through refinements of notions of morphism between algebraic structures. Roughly, morphisms are maps (i.e. functions) that are 'structure-preserving'. The notion is therefore fundamental in all mathematical theories in which the objects of study have some kind of structure, or algebra. The most basic forms of morphism are the *homomorphisms*. These essentially give us a way of embedding a structure (the source) into another one (the target), so that all the relations in the source are present in the target. The converse, however, need not be true; for this, stronger notions of morphism are needed. One such notion is *isomorphism*, which is, however, extremely strong – isomorphic structures must be essentially the same, i.e. 'algebraically identical'. It is a quest for notions in between homomorphism and isomorphism that led to the discovery of bisimulation.

The kind of structures studied in computer science, philosophical logic, and set theory were forms of rooted directed graphs. On such graphs bisimulation is coarser than graph isomorphism because, intuitively, bisimulation allows us to observe a graph only through the movements that are possible along its edges. By contrast, with isomorphisms the identity of the nodes is observable too. For instance, isomorphic graphs have the same number of nodes, which need not be the case for bisimilar graphs (bisimilarity on two graphs indicates that their roots are related in a bisimulation).

The independent discovery of bisimulation in three different fields suggests that only limited exchanges and contacts among researchers existed at the time. The common concept of bisimulation has somehow helped to improve this situation. An example of this are the advances in set theory and computer science derived from Aczel's work.

Bisimilarity and the bisimulation proof method represent examples of a coinductive definition and the coinduction proof method, and as such are intimately related to fixed points, in particular greatest fixed points. We therefore also discuss the introduction of fixed points, and of coinduction. In this case, however, with a more limited breadth: we only consider computer science – fixed points have a much longer history in mathematics – and we simply discuss the main papers in the introduction of coinduction and fixed-point theory in the field. We conclude with some historical remarks on the main fixed-point theorems that underpin the theory of induction and coinduction presented in [San12].

In each section of the chapter, we focus on the origins of the concept dealt with in that section, and do not attempt to follow the subsequent developments. The style of presentation is generally fairly informal, but – we hope – technical enough to make the various contributions clear, so that the reader can appreciate them.

I believe that examining the developments through which certain concepts have come to light and have been discovered is not a matter of purely intellectual curiosity, but it also useful to understand the concepts themselves (e.g. problems and motivations behind them, relationship with other concepts, alternatives, etc.). Further, the chapter offers the opportunity of discussing bisimulation in set theory, where nowadays the role of bisimulation is well recognised. Set theory is not considered in other chapters of the book.

Structure of the chapter In Sections 1.2–1.4 we examine the origins of bisimulation and bisimilarity in modal logic, computer science, and set theory. We report on the introduction of coinduction and fixed points in computer science in Section 1.5, and, in Section 1.6, we discuss the fixed-point theorems.

This chapter is extracted from [San09]; I refer to this for further details on the topic. Solutions to most of the exercises can be found in [San09] and in the Web pages for the book.

Acknowledgements I am very grateful to the following people who helped me to find relevant papers and materials or helped me in tracing back bits of history: L. Aceto, P. Aczel, G. Boudol, J. van Benthem, E. Clarke, Y. Deng, R. Hinnion, F. Honsell, J.-P. Katoen, A. Mazurkiewicz, Y. N. Moschovakis, L.

Moss, R. Milner, U. Montanari, P. Panangaden, W.P. de Roever, W. Thomas, M. Tofte.

1.2 Bisimulation in modal logic

1.2.1 Modal logics

Philosophical logic studies and applies logical techniques to problems of interest to philosophers, somewhat similarly to what mathematical logic does for problems that interest mathematicians. Of course, the problems do not only concern philosophers or mathematicians; for instance, nowadays both philosophical and mathematical logics have deep and important connections with computer science.

Strictly speaking, in philosophical logic a modal logic is any logic that uses *modalities*. A modality is an operator used to qualify the truth of a statement, that is, it creates a new statement that makes an assertion about the truth of the original statement.

For the discussion below we use the 'full propositional logic' introduced in [Sti12], to which we refer for explanations and the interpretation of the formulas:

$$\phi \stackrel{\text{def}}{=} p \;\mid\; \neg\phi \;\mid\; \phi_1 \wedge \phi_2 \;\mid\; \langle\mu\rangle\phi \;\mid\; \bot$$

where p is a proposition letter. We recall that *Labelled Transition Systems (LTSs) with a valuation*, also called *Kripke models*, are the models for the logic. In the examples we will give, when we do not mention proposition letters it is intended that no proposition letters hold at the states under consideration.

1.2.2 From homomorphism to p-morphism

Today, some of the most interesting results in the expressiveness of modal logics rely on the notion of bisimulation. Bisimulation is indeed discovered in modal logic when researchers begin to investigate seriously issues of expressiveness for the logics, in the 1970s. For this, important questions tackled are: When is the truth of a formula preserved when the model changes? Or, even better, under which model constructions are modal formulas invariant? Which properties of models can modal logics express? (When moving from a model $\mathcal{M}$ to another model $\mathcal{N}$, preserving a property means that if the property holds in $\mathcal{M}$ then it holds also when one moves to $\mathcal{N}$; the property being invariant means that also the converse is true, that is, the property holds in $\mathcal{M}$ iff it holds when one moves to $\mathcal{N}$.)

To investigate such questions, it is natural to start from the most basic structure-preserving construction, that of *homomorphism*. A homomorphism from a model $\mathcal{M}$ to a model $\mathcal{N}$ is a function F from the points of $\mathcal{M}$ to the points of $\mathcal{N}$ such that

- whenever a proposition letter holds at a point P of $\mathcal{M}$ then the same letter also holds at $F(P)$ in $\mathcal{N}$;
- whenever there is a μ-transition between two points P, P' in $\mathcal{M}$ then there is also a μ-transition between $F(P)$ and $F(P')$ in $\mathcal{N}$.

Thus, contrasting homomorphism with bisimulation, we note that

(i) homomorphism is a functional, rather than relational, concept;
(ii) in the definition of homomorphism there is no back condition; i.e. the reverse implication, from transitions in $\mathcal{N}$ to those in $\mathcal{M}$, is missing.

It is easy to see that homomorphisms are too weak to respect the truth of modal formulas:

Exercise 1.2.1 Show, by means of a counterexample, that modal formulas are not preserved by homomorphisms. □

That is, a homomorphism H from a model $\mathcal{M}$ to a model $\mathcal{N}$ does not guarantee that if a formula holds at a point Q of $\mathcal{M}$ then the same formula also holds at $H(Q)$ in $\mathcal{N}$.

The culprit for the failure of homomorphisms is the lack of a back condition. We can therefore hope to repair the invariance by adding some form of reverse implication. There are two natural ways of achieving this:

(1) turning the 'implies' of the definition of homomorphism into an 'iff' (that is, a propositional letter holds at P in $\mathcal{M}$ iff it holds at $F(P)$ in $\mathcal{N}$; and $P \xrightarrow{\mu} P'$ in $\mathcal{M}$ iff $F(P) \xrightarrow{\mu} F(P')$ in $\mathcal{N}$, for any P and P');
(2) explicitly adding back conditions (that is, if a propositional letter holds at $F(P)$ in $\mathcal{N}$ then it also holds at P in $\mathcal{M}$; and if in $\mathcal{N}$ there is a transition $F(P) \xrightarrow{\mu} Q$, for some point Q, then in $\mathcal{M}$ there exists a point P' such that $P \xrightarrow{\mu} P'$ and $Q = F(P')$.

Solution (1) is the requirement of *strong homomorphisms*. Solution (2) is first formalised by Krister Segerberg in his famous dissertation [Seg71], as the requirement of *p-morphisms*.

Segerberg starts the study of morphisms between models of modal logics that preserve the truth of formulas in [Seg68]. Initially, p-morphisms are called *pseudo-epimorphims* [Seg68], and are indeed surjective mappings. Later

[Seg70, Seg71], the term is shortened to p-morphisms, and thereafter used to denote also non-surjective mappings. A notion similar to p-morphisms had also occurred earlier, in a work of Jongh and Troelstra [JT66], for certain surjective mappings on partial orders that were called *strongly isotone*. Sometimes, today, p-morphisms are called *bounded morphisms*, after Goldbatt [Gol89]. The p-morphisms can be regarded as the natural notion of homomorphism in LTSs or Kripke models; indeed other reasons make p-morphisms interesting for modal logics, for instance they are useful in the algebraic semantics of modal logics (e.g. when relating modal algebras).

With either of the additions in (1) or (2), the invariance property holds: modal formulas are invariant both under surjective strong homomorphisms and under p-morphisms. (The surjective condition is necessary for strong homomorphisms, but not for p-morphisms.)

Exercise 1.2.2

(1) Exhibit a surjective p-morphism that is not a surjective strong homomorphism.
(2) Show that the invariance property does not hold for non-surjective strong homomorphisms. □

As far as invariance is concerned, the surjective strong homomorphism condition is certainly a very strong requirement – we are not far from isomorphism, in fact (the only difference is injectivity of the function, but even when functions are not injective only states with essentially the 'same' transitions can be collapsed, that is, mapped onto the same point). In contrast, p-morphisms are more interesting. Still, they do not capture all situations of invariance. That is, there can be states s of a model $\mathcal{M}$ and t of a model $\mathcal{N}$ that satisfy exactly the same modal formulas and yet there is no p-morphisms that take s into t or vice versa (Exercise 1.2.3).

1.2.3 Johan van Benthem

The next step is made by Johan van Benthem in his PhD thesis [Ben76] (the book [Ben83] is based on the thesis), who generalises the directional relationship between models in a p-morphism (the fact that a p-morphism is a function) to a symmetric one. This leads to the notion of bisimulation, which van Benthem calls *p-relation*. (Later [Ben84] he renames p-relations as *zigzag relations*.) On Kripke models, a p-relation between models $\mathcal{M}$ and $\mathcal{N}$ is a total relation $\mathcal{S}$ on the states of the models (the domain of $\mathcal{S}$ are the states of $\mathcal{M}$ and the codomain the states of $\mathcal{N}$) such that whenever $P\ \mathcal{S}\ Q$ then: a propositional letter holds at

P iff it holds at Q; for all P' with $P \xrightarrow{\mu} P'$ in $\mathcal{M}$ there is Q' such that $Q \xrightarrow{\mu} Q'$ in $\mathcal{N}$ and $P' \; \mathcal{S} \; Q'$; the converse of the previous condition, on the transitions from Q.

Exercise 1.2.3 Find an example of points in two models that are in a p-relation but no p-morphism can be established between them. □

Van Benthem defines p-relations while working on *correspondence theory*, precisely the relationship between modal and classical logics. Van Benthem's objective is to characterise the fragment of first-order logic that 'corresponds' to modal logic – an important way of measuring expressiveness. He gives a sharp answer to the problem, via a theorem that is today called the 'van Benthem characterisation theorem'. In nowadays's terminology, van Benthem's theorem says that a first-order formula A containing one free variable is equivalent to a modal formula iff A is invariant for bisimulations. That is, modal logic is the fragment of first-order logic whose formulas have one free variable and are invariant for bisimulation. We refer to [Sti12] for discussions on this theorem.

The original proof of the theorem is also interesting. The difficult implication is the one from right to left. A key part of the proof is to show that a point P in a model $\mathcal{M}$ and a point Q in a model $\mathcal{N}$ satisfy the same modal formulas if there are extensions $\mathcal{M}'$ and $\mathcal{N}'$ of the models $\mathcal{M}$ and $\mathcal{N}$ in which P and Q are bisimilar. The extensions are obtained as the limits of appropriate elementary chains of models, starting from the original models. Further, the embedding of the original models into the limits of the chains preserves modal formulas. The reason why it is necessary to move from the original models $\mathcal{M}$ and $\mathcal{N}$ to the extended models $\mathcal{M}'$ and $\mathcal{N}'$ is that on arbitrary models two points may satisfy the same set of formulas without being bisimilar. This may occur if the models are not finitely branching. By contrast, the extended models $\mathcal{M}'$ and $\mathcal{N}'$ are 'saturated', in the sense that they have 'enough points'. On such models, two points satisfy the same modal formulas iff they are bisimilar. As all finitely branching models are saturated, van Benthem's construction also yields the familiar Hennessy–Milner theorem for modal logics [HM85] (an earlier version is [HM80]): on finitely branching models, two points are bisimilar iff they satisfy the same modal formulas. Saturated models need not be finitely branching, however, thus van Benthem's construction is somewhat more general. Note that the need for saturation also would disappear if the logic allowed some infinitary constructions, for instance infinite conjunction. In modern textbooks, such as [BRV01], the proof is sometimes presented in a different way, by directly appealing to the existence of saturated models; however, elementary chains are employed to show the existence of such saturated

models. Again, for details on the above proof and on the characterisation of bisimulation via modal logic, we refer to [Sti12].

After van Benthem's theorem, bisimulation has been used extensively in modal logic, for instance, to analyse the expressive power of various dialects of modal logics, to understand which properties of models can be expressed in modal logics, and to define operations on models that preserve the validity of modal formulas.

1.2.4 Discussion

In philosophical logic we see, historically, the first appearance of the notion of bisimulation. We do not find here, however, coinduction, at least not in an explicit way. Thus total relations between models that represent bisimulations are defined – the p-relations – but there is no explicit definition and use of bisimilarity. Similarly no links are made to fixed-point theory.

In retrospect, today we could say that bisimulation, as a means of characterising equivalence of modal properties, 'was already there' in the *Ehrenfeucht–Fraïssé games*. In the 1950s, Roland Fraïssé [Fra53] gave an algebraic formulation, as a weak form of isomorphism, of indistinguishability by formulas of first-order logic. Andrzej Ehrenfeucht [Ehr61] then extended the result and gave it a more intuitive game-theoretic formulation, in what is now called the Ehrenfeucht–Fraïssé games. Such games are today widely used in computer science, notably in logic, finite model theory, but also in other areas such as complexity theory, following Immerman [Imm82]. It is clear that the restriction of the Ehrenfeucht–Fraïssé games to modal logic leads to game formulations of bisimulation. However, such a connection has been made explicit only after the discovery of bisimulation. See, for instance, Thomas [Tho93].

1.3 Bisimulation in computer science

1.3.1 Algebraic theory of automata

In computer science, the search for the origins of bisimulation takes us back to the algebraic theory of automata, well-established in the 1960s. A good reference is Ginzburg's book [Gin68]. Homomorphisms can be presented on different forms of automata. We follow here *Mealy automata*. In these automata, there are no initial and final states; however, an output is produced whenever an input letter is consumed. Thus Mealy automata can be compared on the set of output strings produced. Formally, a Mealy automaton is a 5-tuple $(W, \Sigma, \Theta, \mathcal{T}, \mathcal{O})$ where

- W is the finite set of *states*;
- Σ is the finite set of *inputs*;
- Θ is a finite set of *outputs*;
- $\mathcal{T}$ is the *transition function*, that is a set of partial functions $\{\mathcal{T}_a \mid a \in \Sigma\}$ from W to W;
- $\mathcal{O}$ is the *output function*, that is, a set of partial functions $\{\mathcal{O}_a \mid a \in \Sigma\}$ from W to Θ.

The output string produced by a Mealy automaton is the *translation* of the input string with which the automaton was fed; of course the translation depends on the state on which the automaton is started. Since transition and output functions of a Mealy automaton are partial, not all input strings are consumed entirely.

Homomorphism is defined on Mealy automata following the standard notion in algebra, e.g. in group theory: a mapping that commutes with the operations defined on the objects of study. Below, if A is an automaton, then W^A is the set of states of A, and similarly for other symbols. As we deal with partial functions, it is convenient to view these as relations, and thereby use for them relational notations. Thus fg is the composition of the two function f and g where f is used first (that is, $(fg)(a) = g(f(a))$); for this, one requires that the codomain of f be included in the domain of g. Similarly, $f \subseteq g$ means that whenever f is defined then so is g, and they give the same result.

A *homomorphism* from the automaton A to the automaton B is a surjective function F from W^A to W^B such that for all $a \in \Sigma$:

(1) $\mathcal{T}_a^A F \subseteq F \mathcal{T}_a^B$ (condition on the states); and
(2) $\mathcal{O}_a^A \subseteq F \mathcal{O}_a^B$ (condition on the outputs).

(We assume here for simplicity that the input and output alphabets are the same, otherwise appropriate coercion functions would be needed.)

At the time (the 1960s), homomorphism and similar notions are all expressed in purely algebraic terms. Today we can make an operational reading of them, which for us is more enlightening. Writing $P \overset{a}{\underset{b}{\rightarrow}} Q$ if the automaton, on state P and input a, produces the output b and evolves into the state Q, and assuming for simplicity that $\mathcal{O}_a^A$ and $\mathcal{T}_a^A$ are undefined exactly on the same points, the two conditions above become:

- for all $P, P' \in W^A$, if $P \overset{a}{\underset{b}{\rightarrow}} P'$ then also $F(P) \overset{a}{\underset{b}{\rightarrow}} F(P')$.

Homomorphisms are used in that period to study a number of properties of automata. For instance, minimality of an automaton becomes the condition that the automaton has no proper homomorphic image. Homomorphisms are also used to compare automata. Mealy automata are compared using the notion

of *covering* (written $\leq$): $A \leq B$ (read 'automaton B covers automaton A') if B can do, statewise, at least all the translations that A does. That is, there is a total function ψ from the states of A to the states of B such that, for all states P of A, all translations performed by A when started in P can also be performed by B when started in $\psi(P)$. Note that B can however have states with a behaviour completely unrelated to that of any state of A; such states of B will not be the image of states of A. If both $A \leq B$ and $B \leq A$ hold, then the two automata are deemed *equivalent*.

Homomorphism implies covering, i.e. if there is a homomorphism from A to B then $A \leq B$. The converse result is (very much) false. The implication becomes stronger if one uses *weak homomorphisms*. These are obtained by relaxing the functional requirement of homomorphism into a relational one. Thus a weak homomorphism is a total relation $\mathcal{R}$ on $W^A \times W^B$ such that for all $a \in \Sigma$:

(1) $\mathcal{R}^{-1}\mathcal{T}_a^A \subseteq \mathcal{T}_a^B\mathcal{R}^{-1}$ (condition on the states); and
(2) $\mathcal{R}^{-1}\mathcal{O}_a^A \subseteq \mathcal{O}_a^B$ (condition on the outputs)

where relational composition, inverse, and inclusion are defined in the usual way for relations [San12, section 0.5], and again functions are taken as special forms of relations. In an operational interpretation as above, the conditions give:

- whenever $P \mathcal{R} Q$ and $P \overset{a}{\underset{b}{\rightarrow}} P'$ hold in A, then there is Q' such that $Q \overset{a}{\underset{b}{\rightarrow}} Q'$ holds in B and $P' \mathcal{R} Q'$.

(On the correspondence between the algebraic and operational definitions, see also Remark 1.3.1 below.) Weak homomorphism reminds us of the notion of simulation (see [San12]). The former is however stronger, because the relation $\mathcal{R}$ is required to be *total*. (Also, in automata theory, the set of states and the sets of input and output symbols are required to be finite, but this difference is less relevant.)

Remark 1.3.1 To understand the relationship between weak homomorphisms and simulations, we can give an algebraic definition of simulation on LTSs, taking these to be triples $(W, \Sigma, \{\mathcal{T}_a \mid a \in \Sigma\})$ whose components have the same interpretation as for automata. A simulation between two LTSs A and B becomes a relation $\mathcal{R}$ on $W^A \times W^B$ such that, for all $a \in \Sigma$, condition (1) of weak homomorphism holds, i.e.

- $\mathcal{R}^{-1}\mathcal{T}_a^A \subseteq \mathcal{T}_a^B\mathcal{R}^{-1}$.

This is precisely the notion of simulation (as defined operationally in [San12]). Indeed, given a state $Q \in W^B$ and a state $P' \in W^A$, we have $Q\,\mathcal{R}^{-1}\,\mathcal{T}_a^A\,P'$ whenever there is $P \in W^A$ such that $P \xrightarrow{a} P'$. Then, requiring that the pair (Q, P') is also in $\mathcal{T}_a^B\mathcal{R}^{-1}$ is the demand that there is Q' such that $Q \xrightarrow{a} Q'$ and $P'\ \mathcal{R}\ Q'$. □

Exercise 1.3.2 Suppose we modified the condition on states of weak homomorphism as follows:

- $\mathcal{T}_a^A\mathcal{R} \subseteq \mathcal{R}\mathcal{T}_a^B$,

and similarly for the condition on outputs. Operationally, what would this mean? What is the relationship to simulations? □

Exercise 1.3.3 Suppose we strengthen the condition in Remark 1.3.1 by turning the inclusion $\subseteq$ into an equality. What would it mean, operationally? Do we obtain bisimulations? Do we obtain relations included in bisimilarity? How can bisimulation (on LTSs) be formulated algebraically? □

As homomorphisms, so weak homomorphisms imply covering. The result for weak homomorphism is stronger as the homomorphisms are strictly included in the weak homomorphisms.

Exercise 1.3.4 Find an example of a weak homomorphism that is not a homomorphism. □

Exercise 1.3.5 Show that there can be automata B and A with $A \leq B$ and yet there is no weak homomorphism between A and B. (Hint: use the fact that the relation of weak homomorphism is total.) □

In conclusion: in the algebraic presentation of automata in the 1960s we find concepts that remind us of bisimulation, or better, simulation. However, there are noticeable differences, as we have outlined above. But the most important difference is due to the fact that the objects are deterministic. To see how significant this is, consider the operational reading of weak homomorphism, namely 'whenever $P\ \mathcal{R}\ Q$ ... then there is Q' such that ....'. As automata are deterministic, the existential in front of Q' does not play a role. Thus the alternation of universal and existential quantifiers – a central aspect of the definitions of bisimulation and simulation – does not really show up on deterministic automata.

1.3.2 Robin Milner

Decisive progress towards bisimulation is made by Robin Milner in the 1970s. Milner transplants the idea of weak homomorphism into the study of the behaviour of programs in a series of papers in the early 1970s ([Mil70, Mil71a, Mil71b], with [Mil71b] being a synthesis of the previous two). He studies programs that are sequential, imperative, and that may not terminate. He works on the comparisons among such programs. The aim is to develop techniques for proving the correctness of programs, and for abstracting from irrelevant details so that it is clear when two programs are realisations of the same algorithm. In short, the objective is to understand when and why two programs can be considered 'intensionally' equivalent.

To this end, Milner proposes – appropriately adapting it to his setting – the algebraic notion of weak homomorphism that we have described in Section 1.3.1. He renames weak homomorphism as *simulation*, a term that better conveys the idea of the application in mind. Although the definition of simulation is still algebraic, Milner now clearly spells out its operational meaning. But perhaps the most important contribution in his papers is the proof technique associated to simulation that he strongly advocates. This techniques amounts to exhibiting the set of pairs of related states, and then checking the simulation clauses on each pair. The strength of the technique is precisely the *locality* of the checks that have to be made, in the sense that we only look at the immediate transitions that emanate from the states (as opposed to, say, trace equivalence where one considers sequences of transitions, which may require examining states other than the initial one of a sequence). The technique is proposed to prove not only results of simulation, but also results of input/output correctness for programs, as a simulation between programs implies appropriate relationships on their inputs and outputs. Besides the algebraic theory of automata, other earlier works that have been influential for Milner are those on program correctness, notably Floyd [Flo67], Manna [Man69], and Landin [Lan69], who pioneers the algebraic approach to programs.

Formally, however, Milner's simulation remains the same as weak homomorphism and as such it is not today's simulation. Programs for Milner are deterministic, with a total transition function, and these hypotheses are essential. Non-deterministic and concurrent programs or, more generally, programs whose computations are trees rather than sequences, are mentioned in the conclusions for future work. It is quite possible that if this challenge had been quickly taken up, then today's notion of simulation (or even bisimulation) would have been discovered much earlier.

Milner himself, later in the 1970s, does study concurrency very intensively, but under a very different perspective: he abandons the view of parallel

programs as objects with an input/output behaviour akin to functions, in favour of the view of parallel programs as *interactive* objects. This leads Milner to develop a new theory of processes and a calculus – CCS – in which the notion of behavioural equivalence between processes is fundamental. Milner however keeps, from his earlier works, the idea of 'locality' – an equivalence should be based outcomes that are local to states.

The behavioural equivalence that Milner puts forward, and that is prominent in the first book on CCS [Mil80], is inductively defined. It is the stratification of bisimilarity $\sim_\omega \stackrel{\text{def}}{=} \bigcap_n \sim_n$ presented in [San12, section 2.10]. Technically, in contrast with weak homomorphisms, $\sim_\omega$ has also the reverse implication (on the transitions of the second components of the pairs in the relation), and can be used on non-deterministic structures. The addition of a reverse implication was not obvious. For instance, a natural alternative would have been to maintain an asymmetric basic definition, possibly refine it, and then take the induced equivalence closure to obtain a symmetric relation (if needed). Indeed, among the main behavioural equivalences in concurrency – there are several of them, see [Gla93, Gla90] – bisimulation is the only one that is not naturally obtained as the equivalence-closure of a preorder.

With Milner's advances, the notion of bisimulation is almost there: it remained to turn an inductive definition into a coinductive one. This will be David Park's contribution.

It is worth pointing out that, towards the end of the 1970s, homomorphisms-like notions appear in other attempts at establishing 'simulations', or even 'equivalences', between concurrent models – usually variants of Petri nets. Good examples are John S. Gourlay, William C. Rounds, and Richard Statman [GRS79] and Kurt Jensen [Jen80], which develop previous work by Daniel Brand [Bra78] and Y.S. Kwong [Kwo77]. Gourlay, Rounds, and Statman's homomorphisms (called *contraction*) relate an abstract system with a more concrete realisation of it – in other words, a specification with an implementation. Jensen's proposal (called *simulation*), which is essentially the same as Kwong's *strict reduction* [Kwo77], is used to compare the expressiveness of different classes of Petri nets. The homomorphisms in both papers are stronger than today's simulation or bisimulation; for instance they are functions rather than relations. Interestingly, in both cases there are forms of 'reverse implications' on the correspondences between the transitions of related states. Thus these homomorphisms, but especially those in [GRS79], remind us of bisimulation, at least in the intuition behind it. In [GRS79] and [Jen80], as well as other similar works of that period, the homomorphisms are put forward because they represent conditions sufficient to preserve certain important properties (such as Church–Rosser and deadlock freedom). In contrast with Milner, little emphasis

is given to the proof technique based on local checks that they bear upon. For instance the definitions of the homomorphisms impose correspondence on *sequences* of actions from related states.

1.3.3 David Park

In 1980, Milner returns to Edinburgh after a six-month appointment at Aarhus University, and completes his first book on CCS. Towards the end of that year, David Park begins a sabbatical in Edinburgh, and stays at the top floor of Milner's house.

Park is one of the top experts in fixed-point theory at the time. He makes the final step in the discovery of bisimulation precisely guided by fixed-point theory. Park notices that the inductive notion of equivalence that Milner is using for his CCS processes is based on a monotone functional over a complete lattice. And by adapting an example by Milner, he sees that Milner's equivalence ($\sim_\omega$) is not a fixed point for the functional, and that therefore the functional is not cocontinuous. He then defines bisimilarity as the greatest fixed point of the functional, and derives the bisimulation proof method from the theory of greatest fixed points. Further, Park knows that, to obtain the greatest fixed point of the functional in an inductive way, the ordinals and transfinite induction, rather then the naturals and standard induction, are needed [San12, theorem 2.8.8]. Milner immediately and enthusiastically adopts Park's proposal. Milner knew that $\sim_\omega$ is not invariant under transitions. Indeed he is not so much struck by the difference between $\sim_\omega$ and bisimilarity as behavioural equivalences, as the processes exhibiting such differences can be considered rather artificial. What excites him is the coinductive proof technique for bisimilarity. Both bisimilarity and $\sim_\omega$ are rooted in the idea of locality, but the coinductive method of bisimilarity further facilitates proofs. In the years to come Milner makes bisimulation popular and the cornerstone of the theory of CCS [Mil89].

In computer science, the standard reference for bisimulation and the bisimulation proof method is Park's paper 'Concurrency on automata and infinite sequences' [Par81a] (one of the most quoted papers in concurrency). However, Park's discovery is only partially reported in [Par81a], whose main topic is a different one, namely omega-regular languages (extensions of regular languages containing also infinite sequences) and operators for fair concurrency. Bisimulation appears at the end, as a secondary contribution, as a proof technique for trace equivalence on automata. Bisimulation is first given on finite automata, but only as a way of introducing the concept on the Büchi-like automata investigated in the paper. Here, bisimulation has additional clauses that make it non-transitive and different from the definition of bisimulation

we know today. Further, bisimilarity and the coinduction proof method are not mentioned in the paper.

Indeed, Park never writes a paper to report on his findings about bisimulation. It is possible that this does not appear to him a contribution important enough to warrant a paper: he considers bisimulation a variant of the earlier notion of simulation by Milner [Mil70, Mil71b]; and it is not in Park's style to write many papers. A good account of Park's discovery of bisimulation and bisimilarity are the summary and the slides of his talk at the 1981 Workshop on the Semantics of Programming Languages [Par81b].

1.3.4 Discussion

In computer science, the move from homomorphism to bisimulation follows a somewhat opposite path with respect to modal logic: first homomorphisms are made relational, then they are made symmetric, by adding a reverse implication.

It remains puzzling why bisimulation has been discovered so late in computer science. For instance, in the 1960s weak homomorphism is well-known in automata theory and, as discussed in Section 1.3.1, this notion is not that far from simulation. Another emblematic example, again from automata theory, is given by the algorithm for minimisation of deterministic automata, already known in the 1950s [Huf54, Moo56] (also related to this is the Myhill–Nerode theorem [Ner58]). The aim of the algorithm is to find an automaton equivalent to a given one but minimal in the number of states. The algorithm proceeds by progressively constructing a relation $\mathcal{S}$ with all pairs of non-equivalent states. It roughly goes as follows. First step (a) below is applied, to initialise $\mathcal{S}$; then step (b), where new pairs are added to $\mathcal{S}$, is iterated until a fixed point is reached, i.e. no further pairs can be added:

(a) For all states P, Q, if P final and Q is not, or vice versa, then $P \mathcal{S} Q$.
(b) For all states P, Q such that $\neg(P \mathcal{S} Q)$: if there is a such that $\mathcal{T}_a(P) \mathcal{S} \mathcal{T}_a(Q)$ then $P \mathcal{S} Q$.

The final relation gives all pairs of non-equivalent states. Then its complement, say $\overline{\mathcal{S}}$, gives the equivalent states. In the minimal automaton, the states in the same equivalence class for $\overline{\mathcal{S}}$ are collapsed into a single state.

The algorithm strongly reminds us of the partition refinement algorithms for computing bisimilarity and for minimisation modulo bisimilarity, discussed in [AIS12]. Indeed, the complement relation $\overline{\mathcal{S}}$ that one wants to find has a natural coinductive definition, as a form of bisimilarity, namely the largest relation $\mathcal{R}$ such that

(1) if $P \mathcal{R} Q$ then either both P and Q are final or neither is;
(2) for each a, if $P \mathcal{R} Q$ then $\mathcal{T}_a(P) \mathcal{R} \mathcal{T}_a(Q)$.

Further, any relation $\mathcal{R}$ that satisfies the conditions (1) and (2) – that is, any bisimulation – only relates pairs of equivalent states and can therefore be used to determine equivalence of specific states.

The above definitions and algorithm are for deterministic automata. Bisimulation would have been interesting also on non-deterministic automata. Although on such automata bisimilarity does not coincide with trace equivalence – the standard equality on automata – at least bisimilarity implies trace equivalence and the algorithms for bisimilarity have a better complexity (P-complete, rather than PSPACE-complete; see [AIS12]).

Lumpability in probability theory An old concept in probability theory that today may be viewed as somehow reminiscent of bisimulation is Kemeny and Snell's *lumpability* [KS60]. A lumping equivalence is a partition of the states of a continuous-time Markov chain. The partition must satisfy certain conditions on probabilities guaranteeing that related states of the partition can be collapsed (i.e. 'lumped') into a single state. These conditions, having to do with sums of probabilities, are rather different from the standard one of bisimulation. (Kemeny and Snell's lumpability roughly corresponds to what today is called bisimulation for continuous-time Markov chains in the special case where there is only one label for transitions.)

The first coinductive definition of behavioural equivalence, as a form of bisimilarity, that takes probabilities into account appears much later, put forward by Larsen and Skou [LS91]. This paper is the initiator of a vast body of work on coinductive methods for probabilistic systems in computer science. Larsen and Skou were not influenced by lumpability. The link with lumpability was in fact noticed much later [Buc94].

In conclusion: in retrospect we can see that Kemeny and Snell's lumpability corresponds to a very special form of bisimulation (continuous-time Markov chains, only one label). However, Kemeny and Snell's lumpability has not contributed to the discovery of coinductive concepts such as bisimulation and bisimilarity.

1.4 Set theory

In mathematics, bisimulation and concepts similar to bisimulation are formulated in the study of properties of extensionality of models. Extensionality guarantees that equal objects cannot be distinguished within the given model.

When the structure of the objects, or the way in which the objects are supposed to be used, are non-trivial, the 'correct' notion of equality may be non-obvious. This is certainly the case for non-well-founded sets, as they are objects with an infinite depth, and indeed most of the developments in set theory towards bisimulation are made in a line of work on the foundations of theories of non-well-founded sets. Bisimulation is derived from the notion of isomorphism (and homomorphism), intuitively with the objective of obtaining relations coarser than isomorphism but still with the guarantee that related sets have 'the same' internal structure.

Bisimulation is first introduced by Forti and Honsell and, independently, by Hinnion, around the same time (the beginning of the 1980s). It is recognised and becomes important with the work of Aczel and Barwise. Some earlier constructions, however, have a clear bisimulation flavour, notably Mirimanoff's isomorphism at the beginning of the twentieth century.

1.4.1 Non-well-founded sets

Non-well-founded sets are, intuitively, sets that are allowed to contain themselves. As such they violate the *axiom of foundation*, according to which the membership relation on sets does not give rise to infinite descending sequences

$$\ldots A_n \in A_{n-1} \in \ldots \in A_1 \in A_0 \ .$$

For instance, a set Ω which satisfies the equation $\Omega = \{\Omega\}$ is circular and as such non-well-founded. A set can also be non-well-founded without being circular; this can happen if there is an infinite membership chain through a sequence of sets all different from each other.

If the axiom of foundation is used, the sets are *well-founded*. On well-founded sets the notion of equality is expressed by Zermelo's *extensionality axiom*: two sets are equal if they have exactly the same elements. In other words, a set is precisely determined by its elements. This is very intuitive and naturally allows us to reason on equality proceeding by (transfinite) induction on the membership relation. For instance, we can thus establish that the relation of equality is unique. Non-well-founded sets, by contrast, may be infinite in depth, and therefore inductive arguments may not be applicable. For instance, consider the sets A and B defined via the equations $A = \{B\}$ and $B = \{A\}$. If we try to establish that they are equal via the extensionality axiom we end up with a tautology ('A and B are equal iff A and B are equal') that takes us nowhere.

Different formulations of equality on non-well-founded sets appear during the twentieth century, together with proposals for *axioms of anti-foundation*.

1.4.2 The stratified approach to set theory

The first axiomatisation of set theory by Ernst Zermelo in 1908 [Zer08] has seven axioms, among which is the axiom of extensionality. However, it has no axioms of foundation, and the possibility of having circular sets is in fact left open (page 263, op. cit.).

In the same years, Bertrand Russell strongly rejects all definitions that can involve forms of circularity ('whatever involves all of a collection must not be one of the collection', in one of Russell's formulations [Rus08]). He favours a *theory of types* that only allows *stratified* constructions, where objects are hereditarily constructed, starting from atoms or primitive objects at the bottom and then iteratively moving upward through the composite objects. A preliminary version of the theory is announced by Russell already in 1903 [Rus03, Appendix B]; more complete and mature treatments appear in 1908 [Rus08] and later, in 1910, 1912, 1913, in the monumental work with Alfred North Whitehead [RW13].

Russell's approach is followed by the main logicians of the first half of the twentieth century, including Zermelo himself, Abraham Fraenkel, Thoralf Skolem, Johann von Neumann, Kurt Gödel, Paul Bernays. Their major achievements include the formulation of the axiom of foundation, and the proofs of its consistency and independence. An axiom of foundation is deemed necessary so as to have a 'canonical' universe of sets. Without foundation, different interpretations are possible, some including circular sets. This possibility is clearly pointed out as a weakness by Skolem [Sko23], and by Fraenkel [Fra22], where circular sets (precisely, Mirimanoff's 'ensembles extraordinaires', see below) are labelled as 'superfluous'. It will be formally proved by Bernays only in 1954 [Ber54] that the existence of circular sets does not lead to contradictions in the Zermelo–Fraenkel system without the axiom of foundation.

Remark 1.4.1 The axiom of foundation forces the universe of sets in which the other axioms (the basic axioms) should be interpreted to be the smallest possible one; i.e. to be an 'inductive universe'. By contrast, axioms of anti-foundation lead to the largest possible universe, i.e. a 'coinductive universe'. Indeed, referring to the algebraic/coalgebraic interpretation of induction/coinduction, the foundation axiom can be expressed as a requirement that the universe of sets should be an *initial algebra* for a certain powerset functor, whereas anti-foundation (as in Forti and Honsell, Aczel, and Barwise) can be expressed as a requirement that the universe should be a *final coalgebra* for the same functor. The former is an inductive definition of the universe, whereas the latter is a coinductive one. □

The motivations for formalising and studying the stratified approach advocated by Russell were strong at the beginning of the twentieth century. The discovery of paradoxes such as Burali-Forti's and Russell's had made the set theory studied by Cantor and Frege shaky, and circularity – with no distinction of cases – was generally perceived as the culprit for these as well as for paradoxes known in other fields. Further, the stratified approach was in line with common sense and perception (very important in Russell's conception of science), which denies the existence of circular objects.

The stratified approach remains indeed *the only* approach considered (in logics and set theory), up to roughly the 1960s, with the exception of Mirimanoff and Finsler that we discuss below. The stratified approach has also inspired – both in the name and in the method – type theory in computer science, notably in the works of Church, Scott, and Martin-Löf. It will be first disputed by Jean-Yves Girard and John Reynolds, in the 1970s, with the introduction of impredicative polymorphism.

1.4.3 Non-well-founded sets and extensionality

Dimitry Mirimanoff first introduces in 1917 the distinction between well-founded and non-well-founded sets, the 'ensembles *ordinaires* et *extraordinaires*' in Mirimanoff's words [Mir17a] (on the same topic are also the two successive papers [Mir17b] and [Mir20]). Mirimanoff realises that Zermelo's set theory admitted sophisticated patterns of non-well-foundedness, beyond the 'simple' circularities given by self-membership as in the purely reflexive set $\Omega = \{\Omega\}$. In [Mir17b], Mirimanoff also tries to give an intuition for the non-well-founded sets; he recalls the cover of a children's book he had seen, with the image of two children looking at the cover of a book, which in turn had the image of two children, in a supposedly infinite chain of nested images.

Mirimanoff defines an interesting notion of isomorphism between sets, that we report in Section 1.4.8. Mirimanoff does not however go as far as proposing an axiom of extensionality more powerful than Zermelo's. This is first attempted by Paul Finsler, in 1926 [Fin26]. Finsler presents three axioms for a universe of sets equipped with the membership relation. The second one is an extensionality axiom, stipulating that isomorphic sets are equal. Finsler's notion of isomorphism between two sets X and Y – which is different from Mirimanoff's – is, approximately, a bijection between the transitive closures of X and Y (more precisely, the transitive closures of the unit sets $\{X\}$ and $\{Y\}$; the precise meaning of isomorphism for Finsler can actually be debated, for it

appears in different forms in his works).[1] Finsler uses graph theory to explain the properties and structure of sets, something that later Aczel will make more rigorous and at the heart of his theory of non-well-founded sets.

Mirimanoff's and Finsler's works are remarkable: they go against the standard approach to set theory at the time; and against common sense according to which objects are stratified and circular sets are 'paradoxical'. For Mirimanoff and Finsler, not all circular definitions are dangerous, and the challenge is to isolate the 'good' ones.

The attempts by Mirimanoff and Finsler remain little known. We have to wait till around the 1960s with, e.g. Specker [Spe57] and Scott [Sco60], to see a timid revival of interest in non-well-founded structures, and the late 1960s, and then the 1970s and 1980s, for a wider revival, with Boffa (with a number of papers, including [Bof68, Bof69, Bof72]) and many others. New proposals for anti-foundation axioms are thus made, and with them, new interpretations of extensionality on non-well-founded sets, notably from Scott [Sco60], and Forti and Honsell [FH83]. Forti and Honsell obtain bisimulation; their work is then developed by Aczel and Barwise. We discuss the contributions of Forti and Honsell, Aczel, and Barwise's. On the history of non-well-founded sets, the reader may also consult Aczel [Acz88, appendix A].

1.4.4 Marco Forti and Furio Honsell

Marco Forti's and Furio Honsell's work on non-well-founded sets [Hon81, FH83] (and various papers thereafter) is spurred by Ennio De Giorgi, a well-known analyst who, in the 1970s and 1980s, organises regular weekly meetings at the Scuola Normale Superiore di Pisa, on logics and foundations of Mathematics. In some of these meetings, De Giorgi proposes constructions that could yield infinite descending chains of membership on sets, that Forti and Honsell then go on to elaborate and develop.

The most important paper is [FH83]. Here Forti and Honsell study a number of anti-foundation axioms, derived from a 'Free Construction Principle' proposed by De Giorgi. They include axioms that already appeared in the literature (such as Scott's [Sco60]), and a new one, called X_1, that gives the strongest extensionality properties, in the sense that it equates more sets. (We recall X_1 in the next section, together with Aczel's version of it.) The main objective of the

[1] A set A is transitive if each set B that is an element of A has the property that all the elements of B also belong to A; that is, all composite elements of A are also subsets of A. The transitive closure of a set C is the smallest transitive set that contains C. Given C, its transitive closure is intuitively obtained by copying at the top level all sets that are elements of C, and then recursively continuing so with the new top-level sets.