Lecture Notes in Computer Science 1426
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Frédéric Geurts

Abstract Compositional Analysis of Iterated Relations

A Structural Approach
to Complex State Transition Systems

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Author

Frédéric Geurts
Service de Mathématiques de la Gestion
Université Libre de Bruxelles
CP 210 01, Boulevard du Triomphe
B-1050 Bruxelles, Belgium
E-mail: fgeurts@smg.ulb.ac.be

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Geurts, Frédéric:
Abstract compositional analysis of iterated relations : a structural approach to complex state transition systems / Frédéric Geurts. - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1998
(Lecture notes in computer science ; Vol. 1426)
ISBN 3-540-65506-9

CR Subject Classification (1998): F.1, F.3.1, C.3, D.2.4

ISSN 0302-9743
ISBN 3-540-65506-9 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10637524 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Foreword by Michel Sintzoff

The present book apparently falls outside of the scope of the LNCS series: the theory of dynamical systems is mainly used for systems defined by, say, differential equations, and very little for programs. Yet, to consider programs as dynamical systems sheds light at least on the relationship between discrete-time systems and continuous-time ones; this is an important issue in the area of hybrid systems, where control engineers and software designers learned to work hand in hand.

As a matter of fact, program traces constitute time-to-state functions, and programs which define sets of traces characterize reactive systems as used in industry and services. Quite similarly, differential systems define sets of time-to-state functions, and they serve in many disciplines, e.g. physics, engineering, biology, and economics. Thus, we must relate programs as well as differential equations to dynamical systems.

The concepts of invariance and attraction are central to the understanding of dynamical systems. In the case of programs, we use the quite similar notions of invariance, viz. safety, and reachability, viz. termination or liveness; reachability amounts to finite-time attraction and weakest preconditions determine largest basins of reachability. Accordingly, the basic programming concepts of fairness, fault-tolerance and self-stabilization correspond, in the case of dynamical systems, to recurrence (repeated return to desired states), structural stability (return to desired dynamics after system perturbation), and absorption (return to a desired invariant after state perturbation).

Linear dynamical systems are usually analyzed in terms of analytical expressions which provide explicit solutions for simple differential or difference equations. In the case of nonlinear dynamical systems, exact solutions cannot be obtained in general, and the qualitative analysis is then carried out on the system specifications themselves, viz. on differential equations. For instance, attraction is proven using an energy-like function: the successive dynamical states are abstracted to decreasing non-negative reals. Also, the qualitative analysis of concrete dynamics can be reduced to that of symbolic ones, in which each state is a symbol abstracting a set of concrete states; this shows discrete dynamics can serve as qualitative abstractions of continuous ones.

Similarly to nonlinear systems, programs in general cannot be understood in terms of analytical solutions. Weakest preconditions often become too com-

plex, and practical reasoning methods apply on the programs themselves. For example, invariance is checked by structural induction, and termination is verified using an energy-like function from the successive dynamical states to decreasing non-negative integers. Moreover, the verification of a concrete program, very much as in the case of a concrete nonlinear dynamical system, is better carried out in terms of an abstract, simpler one. This paradigm of abstraction underlies many useful techniques in mathematics as well as in computing; let us recall automata simulation, data representation, abstract interpretation, and time abstraction.

Interestingly enough, the mathematical theory of dynamical systems not only supports abstraction-based methods, e.g. symbolic dynamics, but also introduces basic compositional techniques such as sequential and iterative composition. What could then computing science contribute to that theory? The answer is clear: *scaling up*. Actually, the central results in the classical theory of dynamical systems concern single-level individual systems. For us, the main challenge is to design systems for many complementary goals and at various abstraction levels. To this end, we intensively use the principles of modular composition and stepwise refinement. The same approach could give rise to possible original contributions of computing science in the area of dynamical systems. Indeed, the present book shows how to construct complex dynamics by a systematic composition of simple ones, and thus provides a roadmap to compositional design techniques for scaled-up dynamical systems.

Programming theory has taken great advantage of logic and algebra. It should similarly benefit from the theory of dynamical systems; this synergy would entail a common scientific platform for system engineering at large, including software engineering. Examples of such cross-fertilization already exist. Discrete-event control systems and hybrid systems, combining continuous and discrete time, are specified, analyzed, and synthesized using finite-state automata. Synchronization of dynamics provides a means of secure communication. Emergent computations can be implemented by cellular neural networks. Distributed dynamics help to analyze agent-based systems.

The nice matching between dynamics and computational intuitions explains the success of automata-based requirements, dynamics-based architectures, state-based specifications, object-oriented systems, proof dynamics, and design-process models. At each abstraction level, dynamics can be specified at will using programs, automata, logic, algebra, or calculus. For many-sided and multi-level systems such as the web or a house, the crucial issues are the choice of the right level of dynamics, the interaction of internal dynamics with partially defined external ones, and the scaling-up of state-, control- and time-refinements.

The author must be thanked warmly for providing us with many stimulating ideas on these attractive themes.

Preface

State-transition systems model machines, programs, and specifications [20, 23, 284, 329], but also the growth and decline of ant populations, financial markets, diseases and crystals [22, 35, 178, 209, 279]. In the last decade, the growing use of digital controllers in various environments has entailed the convergence of control theory and real-time systems toward hybrid systems [16] by combining both discrete-event facets of reality with Nature's continuous-time aspects. The computing scientist and the mathematician have re-discovered each other. Indeed, in the late sixties, the programming language Simula, "father" of modern object-oriented languages, had already been specifically designed to model dynamical systems [76].

Today, the importance of computer-based systems in banks, telecommunication systems, TVs, planes and cars results in larger and increasingly complex models. Two techniques had to be developed and are now fruitfully used to keep analytic and synthetic processes feasible: composition and abstraction. A compositional approach builds systems by composing subsystems that are smaller and more easily understood or built. Abstraction simplifies unimportant matters and puts the emphasis on crucial parameters of systems.

In order to deal with the complexity of some state-transition systems and to better understand complex or chaotic phenomena emerging out of the behavior of some dynamical systems, the aim of this monograph is to present first steps toward the integrated study of composition and abstraction in dynamical systems defined by iterated relations.

The main insights and results of this work concern a structural form of complexity obtained by composition of simple interacting systems presenting opposed attracting behaviors. This complexity expresses itself in the evolution of composed systems, i.e., their dynamics, and in the relations between their initial and final states, i.e., the computations they realize. The theoretical results presented in the monograph are then validated by the analysis of dynamical and computational properties of low-dimensional prototypes of chaotic systems (e.g. Smale horseshoe map, Cantor relation, logistic map), high-dimensional spatiotemporally complex systems (e.g. cellular automata), and formal systems (e.g. paperfoldings, Turing machines).

Acknowledgements. This monograph is a revision of my PhD thesis which was completed at the Université catholique de Louvain (Belgium) in March 96.

The results presented here have been influenced by many people and I would like to take this opportunity to thank them all.

In particular, I express my deepest gratefulness to my advisor, Michel Sintzoff, with whom I had the rewarding privilege to collaborate. His generous support, his never-ending interest in my work, his incredibly long-term scientific perspective, and his matchless sense of humour incited me to develop and write things I would never have dreamt of. I owe Michel an abstract compositional virus that flies in the Garden of Structural Similarities.

My gratitude further goes to Yves Kamp, André Arnold, and Michel Verleysen, for their careful reading of draft versions of this text, and for their kind and constructive way to turn simple statements into convincing ones. I am also thankful to Nicola Santoro and Paola Flocchini for their constant belief in my research on cellular automata, and for their multiple invitations to Ottawa.

I wish to thank the staff of the Computer Science Department at UCL, and especially its chairman, Elie Milgrom, for providing the nice environment in which I could spend five exciting years.

I acknowledge the financial support I received from the *Fonds National de la Recherche Scientifique*, the *European Community*, the *Communauté Française de Belgique*, and the *Académie Royale de Belgique*.

My warmest thanks go to my friends Bruno Charlier, Luc Meurant, and Luc Onana Alima, for their irreplaceable presence, and to my parents and sister, Pol, Rose-Marie, and Muriel, for their eternal love, care, and attention.

At last but not least, words are not strong enough to tell my love to my wife, Cécile, and to our beautiful smiling daughter, Romane. Without their emotional support, all this would not have been possible.

I had a dream.
I was there, under the sun,
Waiting for nothing, for happiness.
Quelque chose attira mon attention.
Etait-ce cet oiseau qui volait vers moi ?
Il y avait tant de monde que j'avais peine à distinguer
D'où venait cette douce magie qui m'enrobait.
Puis des notes, une musique sublime, se dévoilèrent,
Et tu apparus, Vénus, d'un océan de joie,
Enivrant de ta douceur bleue le ciel et tous ses astres.

Frédéric Geurts
Louvain-la-Neuve, Belgium
January 1998

Table of Contents