ECML 2007 PKDD
WARSAW POLAND

THE 18TH EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
THE 11TH EUROPEAN CONFERENCE ON PRINCIPLES AND PRACTICE
OF KNOWLEDGE DISCOVERY IN DATABASES

PROCEEDINGS OF THE WORKSHOP ON APPROACHES AND APPLICATIONS OF INDUCTIVE PROGRAMMING

AAIP'07

September 17, 2007

Warsaw, Poland

Editors:
Emanuel Kitzelmann
Ute Schmid
Faculty of Information Systems and Applied Computer Sciences
University of Bamberg
Germany

Preface

The ECML/PKDD workshop AAIP 2007—"Approaches and Applications of Inductive Programming"—was held at the 18th European Conference on Machine Learning (ECML) and the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD) in Warsaw, Germany on 17 September, 2007. The main goal of AAIP was to bring together researchers working on different approaches to inductive programming as well as researchers from different areas of machine learning who are especially interested in the inductive synthesis of programs.

Inductive programming (or inductive program synthesis) incorporates all approaches which are concerned with learning programs or algorithms from incomplete specifications such as, e.g., input/output examples describing the desired behavior of the intended program or traces describing steps of computation of specific outputs. Typical additional input to an IP system is background knowledge such as predefined functions and predicates to be used or schemas/templates defining the general data flow of the intended program. Output of an IP system is a program in some arbitrary programming language containing conditionals and loop- or recursive control structures.

Inductive programming is a research topic of crucial interest for machine learning and artificial intelligence in general. The ability to generalize a program—containing control structures as recursion or loops—from examples is a challenging problem which calls for approaches going beyond the requirements of algorithms for concept learning. Pushing research forward in this area can give important insights in the nature and complexity of learning as well as enlarging the field of possible applications.

Typical application areas where learning of programs or recursive rules are called for, are first in the domain of software engineering where structural learning, software assistants and software agents can help to relieve programmers from routine tasks, give programming support for endusers, or support of novice programmers and programming tutor systems. Further areas of application are language learning, learning recursive control rules for AI-planning, learning recursive concepts in web-mining or for data-format transformations.

Today, different inductive programming algorithms are developed in and use techniques from different research fields, these are evolutionary computation, inductive logic programming, classical (Summers-like) inductive synthesis of functional programs, and grammar inference. They can be clustered into two major approaches: (i) the analytical/example-driven approach where programs are derived by analyzing given examples and (ii) the search-based/generate-and-test approach where programs are enumerated heuristically and then tested against given examples. Both approaches have different strengths and limits. The papers in these proceedings cover both approaches.

The AAIP 2007 workshop brought together researchers from different communities considering inductive programming from different perspectives and with the common interest on induction of general programs regarding theory, methodology and applications. The two invited speakers represent inductive program synthesis research in the areas of evolutionary algorithm design (Roland Olsson, Østfold University College, Norway) and algebraic specification (Lutz Hamel, University of Rhode Island, USA).

We want to thank all those who contributed to AAIP 2007. We thank all members of the program committee for their support in promoting the workshop and for reviewing submitted papers and we thank the ECML organizers, especially the workshop chair Marzena Kryszkiewicz for technical support.

Bamberg, July 2007

Emanuel Kitzelmann
Ute Schmid

Workshop Organization

AAIP Workshop Chairs

Emanuel Kitzelmann (University of Bamberg, Germany)
Ute Schmid (University of Bamberg, Germany)

ECML/PKDD Workshop Chair

Marzena Kryszkiewicz (Warsaw University of Technology, Poland)

AAIP Workshop Program Committee

Ricardo Aler Mur (Universidad Carlos III de Madrid, Spain)
Pierre Flener (Sabanci University, Turkey, and Uppsala University, Sweden)
Emanuel Kitzelmann
Donato Malerba (Universita degli Studi di Bari, Italy)
Stephen Muggleton (Imperial College, UK)
Ute Schmid

Table of Contents

Automatic Design of Algorithms through Evolution (ADATE)

Roland Olsson

Faculty of Computer Science, Østfold University College, Norway
roland.olsson@hiof.no
http://www-ia.hiof.no/˜rolando/

Automatic Design of Algorithms through Evolution (ADATE) is a system for automatic functional programming that is able to synthesize recursive programs with automatic invention of recursive help functions.

The invited talk will first present two recent applications of ADATE where it appears to be highly competitive with the best known alternative methods. The first application is synthesis of programs that drive an autonomous vehicle given input from gyros and range sensors. In the second application, ADATE generates algorithms for image segmentation, that is separating a possibly noisy image into regions representing objects of interest.

The autonomous driving example shows that ADATE is suitable for reinforcement learning and does not need explicitly provided outputs in order to generate desirable programs.

We will explain the basic program transformations employed by ADATE as well as briefly discuss the combinatorial search algorithms needed to efficiently and effectively search for suitable transformation combinations. The talk will also show the population management of ADATE and how it considers both the time complexity of synthesized programs and the need for syntactic complexity minimization in order to avoid overfitting.

Full article in PDF

An Inductive Programming Approach to Algebraic Specification

Lutz Hamel and Chi Shen

Department of Computer Science and Statistics
University of Rhode Island
Kingston, RI 02881, USA
hamel@cs.uri.edu, shenc@cs.uri.edu

Abstract. Inductive machine learning suggests an alternative approach to the algebraic specification of software systems: rather than using test cases to validate an existing specification we use the test cases to induce a specification. In the algebraic setting test cases are ground equations that represent specific aspects of the desired system behavior or, in the case of negative test cases, represent specific behavior that is to be excluded from the system. We call this inductive equational logic programming. We have developed an algebraic semantics for inductive equational logic programming where hypotheses are cones over specification diagrams. The induction of a hypothesis or specification can then be viewed as a search problem in the category of cones over a specific specification diagram for a cone that satisfies some pragmatic criteria such as being as general as possible. We have implemented such an induction system in the functional part of the Maudespecification language using evolutionary computation as a search strategy.

Full article in PDF

Data-Driven Induction of Recursive Functions from Input/Output-Examples

Emanuel Kitzelmann

Faculty of Information Systems and Applied Computer Sciences,
University of Bamberg, 96045 Bamberg, Germany
`emanuel.kitzelmann@wiai.uni-bamberg.de`
`http://www.cogsys.wiai.uni-bamberg.de/kitzelmann/`

Abstract. We describe a technique for inducing recursive functional programs over algebraic datatypes from few non-recursive and only positive ground example-equations. Induction is data-driven and based on structural regularities between example terms. In our approach, functional programs are represented as constructor term rewriting systems containing recursive rewrite rules. In addition to the examples for the target functions, background knowledge functions that may be called by the induced functions can be given in form of ground equations. Our algorithm induces several dependent recursive target functions over arbitrary user-defined algebraic datatypes in one step and automatically introduces auxiliary subfunctions if needed. We have implemented a prototype of the described method and applied it to a number of problems.

Full article in PDF

A Functional Approach to Evolving Recursive Programs

Martin Dostál

Department of Computer Science, Palacky University, Olomouc, Czech Republic
dostal@inf.upol.cz

Abstract. This paper introduces a multi objective function based approach for evolving programs with emphasis on modularity and repetitive code execution. A functional paradigm based programming language for evolution of programs is presented. The evolution of recursive functions using code structure abstraction is discussed henceforward and a kind of structure abstracting functions (AR-functions) is introduced. Some examples and results are presented.

Full article in PDF

Detecting Data Structures from Traces

Alon Itai and Michael Slavkin

Department of Computer Science
Technion Israel Institute of Technology
Haifa, Israel
itai@cs.technion.ac.il, mishaslavkin@yahoo.com

Abstract. This paper shows how to deduce the data structure that gave rise to a trace of memory accesses. The proposed methodology is general enough to classify different sets of data structures and is based on machine learning techniques.

Full article in PDF

Author Index