ISR 2017: Introductory Course

Aart Middeldorp and Sarah Winkler

Abstract

Term rewriting is a conceptually simple, but powerful abstract model of computation which underlies much of declarative programming and automated theorem proving. The foundation of rewriting is equational logic. It constitutes a basic framework for program analysis and has applications in automated reasoning, algebra, computability theory and lambda calculus, compiler construction, engineering, as well as functional and logic programming.

This course provides a modern introduction to rewriting in general and term rewriting in particular. All key concepts are covered and glimpses into current research will be provided. Moreover, several automatic tools will be demonstrated.

Contents

After presenting some motivation examples and basic no(ta)tions for terms and substitutions, abstract rewrite systems (ARSs) are introduced. Crucial properties such as termination, confluence, unique normal forms, and completeness are discussed, and the relationships among them such as Newman's lemma are explained. Ubiquitous proof techniques in rewriting like multiset orders are introduced.

Afterwards the focus is moved to the concrete setting of term rewrite systems (TRSs). The foundations are laid by presenting equational reasoning and notions related to term algebras. Key properties of term rewrite systems such as termination and confluence are shown to be undecidable. By restricting to ground TRSs these properties become decidable. The important congruence closure algorithm is introduced to decide the validity problem in ground systems.

Next, three thematic blocks are discussed in detail. The first in-depth topic is termination. We present several different methods to establish termination. These include well-founded monotone algebras over different domains (in particular polynomial interpretations), the lexicographic path order, and the Knuth-Bendix order. The strengths of the different methods are highlighted by many examples and demonstrated using automatic termination tools. Simple termination and Kruskal's tree theorem are introduced to emphasize the termination hierarchy. Dependency pairs are presented as a powerful technique to establish termination automatically.

In the second block a connection to theorem proving is drawn by delving into Knuth-Bendix completion. After introducing unification and critical pairs as basic tools, a simple completion procedure is presented. The elegant theory

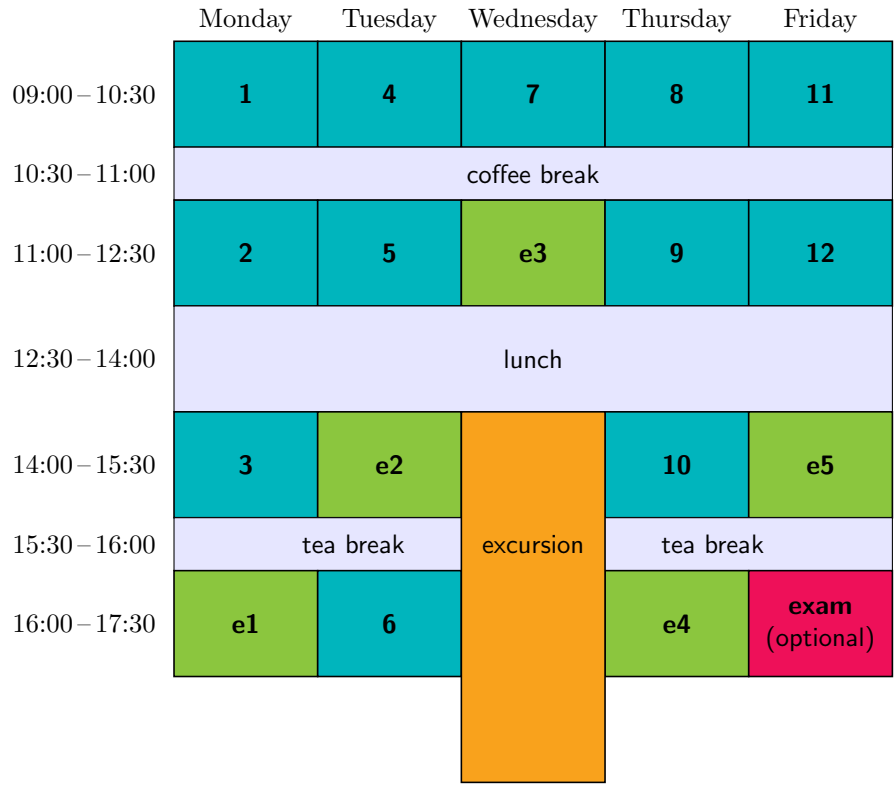
of abstract completion is explained. Examples and completion tools are used to illustrate the developments.

The third block is devoted to confluence and evaluation strategies. Orthogonality is introduced as an easy syntactic criterion that ensures confluence. Common strategies such as call by value and call by need are presented. Strategy annotations which give the user explicit control over the evaluation strategy are also introduced. Advantages and drawbacks of different strategies are emphasized by discussing important properties such as normalization and optimality.

The course concludes by providing an outlook into more advanced and current research topics.

Organization

The material is presented in 12 lecture slots, 5 exercise sessions, and an optional exam:



Prerequisites

The course does not assume any particular prerequisites, except some acquaintance with basic formal methods. A similar course is taught annually at the University of Innsbruck in the bachelor program of computer science.

Literature

The slides, exercise sheets, and course notes will be made available from

http://cl-informatik.uibk.ac.at/users/ami/17isr/

Lecturers

- Aart Middeldorp is professor at the University of Innsbruck and a leading researcher in term rewriting. He has taught courses on term rewriting world-wide for more than 25 years.
- Sarah Winkler is a postdoctoral researcher at the University of Innsbruck. She obtained her doctoral degree in 2013 on a topic in completion and is the main developer of the powerful completion tool mkbTT.