Tutorial Presentations at the Twelfth International Conference on Principles of Knowledge Representation and Reasoning

Leonardo de Moura
Microsoft Research

Carsten Lutz
University of Bremen

monica mc schraefel
University of Southampton

Bernhard Nebel
University of Freiburg

Satisfiability with and without Theories

Tutorial by Leonardo de Moura

Constraint satisfaction problems arise in many diverse areas including software and hardware verification, type inference, extended static checking, test-case generation, scheduling, planning, graph problems, among others. The most well-known constraint satisfaction problem is propositional satisfiability SAT. Of particular recent interest is satisfiability modulo theories (SMT), where the interpretation of some symbols is constrained by a background theory. For example, the theory of arithmetic restricts the interpretation of symbols such as: +, $\leq$, 0, and 1.

SMT draws on the most prolific problems in the past century of symbolic logic: the decision problem, completeness and incompleteness of logical theories, and finally complexity theory. In this tutorial, I will describe a brief introduction to the theory behind SAT and SMT solvers, the main techniques, and their many applications. In particular, I will describe how these solvers are used at Microsoft.

Reasoning in Description Logics: Expressive Power Versus Computational Complexity

Tutorial by Carsten Lutz

Description logics (DLs) are a popular family of knowledge representation formalisms that play an important role as a logical foundation of ontology languages such as OWL and aim at providing a good compromise between representational capabilities (that is, expressive power) and the computational complexity of reasoning. The fact that different applications require different such compromises has led to the development of a large toolbox of DLs that range from inexpressive but computationally efficient to very expressive but computationally challenging, thus catering for various needs and requirements.

This tutorial provides a comprehensive tour of reasoning in modern description logics with an emphasis on computational complexity. It covers expressive DLs such as the SHIQ family, explaining which kind of expressive power tends to make reasoning more complex, and why. It also covers inexpressive DLs such as the EL and DL-Lite families, explaining their good computational behavior and highlighting the limits of polytime reasoning. On top of the traditional reasoning problems such as subsumption and satisfiability, the tutorial also addresses query answering over instance data in the presence of DL ontologies, a more recent reasoning problem that rapidly gains importance in applications. In particular, I will explain how the complexity landscape differs for traditional reasoning and for query answering, and take a brief look at computational complexity issues raised by implementations of DL query answering based on standard relational database systems. Throughout the tutorial, connections to the W3C-standard OWL are drawn whenever possible.

What If You Wanted Someone (Else) to Use This? Usability Heuristics for KR Tool and Representation Design

Tutorial by monica mc schraefel

The user interface or user experience of a tool in KR may be one of the last things on a researcher's mind when trying to find a path towards exploring a problem or automating components to build a solution. And that's fine if the author of the tool is the only person who will ever use the tool, but what if there's a desire to share the tool with someone else? There are some straight ahead heuristics about designing both the interaction with tool components and with the representations that a tool produces to help with usability for domain experts and potentially even legibility of what the tool produces for interested persons outside the domain. There are likewise some lightweight mechanisms to evaluate whether the proposed design will achieve either of these ends before the code is implemented.

Computational Complexity of Action Planning

Tutorial by Bernhard Nebel

After a brief introduction of different planning frameworks and a short introduction to computational complexity theory, we will examine computational complexity results for different planning frameworks and planning domains. In order to get an idea of how changes in expressiveness can affect the applicability of planning methods, we will also have a look at compilation techniques that allow us to assess the expressive power of a planning formalism.