The Rome Workshop of Artificial Intelligence

March 26–28, 2007

Alexander Kleiner

Firemen won't loop

The success of disaster mitigation, i.e. the efficiency of a team of first responders searching for victims after a disaster, depends on their ability to coordinate and thus to be always aware of their location. However, to localize within collapsed buildings without visibility due to smoke and fire is even for humans a challenging task. One solution to this problem is to equip firemen with assistance systems, such as wearable devices performing Simultaneous Localization And Mapping (SLAM) in the background. SLAM methods work after the principle of map improvement through loop-closure, i.e. to improve the map globally each time places have been re-observed. However, facing the reality of emergency response, fireman won't loop.

In this talk I present a solution to this problem based on non-selfish information sharing via the deployment of RFID technology. Furthermore, preliminary results from experiments with pedestrians are shown and discussed.

Dapeng Zhang

Learning a New Action Sequence with a Table Soccer Robot by Observing and Imitating

Star-Kick is a commercially available and fully automatic table soccer robot, which plays table soccer games with human players on a competitive level. With Star-Kick, learning skillful actions similar to a human player in the games demands well-tuned parameters and a not too tedious learning process. Two independent learning algorithms are therefore employed to learn a new "lock" and "slide-kick" action sequence by observing the performed actions and imitating the relative actions of a human player. The experiments with a Star-Kick show that an effective action sequence can be learned in around 20 trials.

Robert Mattmüller

Integrating Selective and Symbolic Approaches for ATL Model Checking

Model-checking Alternating-time temporal logic (ATL) formulae can be accomplished in various different ways. The approach followed in the MOCHA tool is a symbolic BDD-based one; alternatively, one can reduce any ATL model checking instance to a weak game, which can in turn be solved by heuristic game graph search.

Whereas the symbolic approach is most powerful in verifying formulae for which almost the complete state space has to be explored and heuristic guidance is of no use anyway, the explicit heuristic game graph search outperforms symbolic search if only a small fragment of the search space has to be explored.

We will investigate ways of integrating the approaches and compare the results to those of the basic algorithms.

Gabi Röger

Basic Action Theories with the same Expressive Power as ADL

The main focus in the area of action languages such as Golog was put on expressive power, while the development in the area of action planning was focused on efficient plan generation. In recent years one can observe some convergence in the expressive power of PDDL – the language that is used in the field of action planning – and Golog. This brings up the idea of integrating the concepts of both areas, wich would provide great advantages: A user could constrain a system's behaviour on a high level using Golog, while the actual low-level actions are planned by an efficient planning system. First endevours have been made by Eyerich et al. by identifying a subset of the situation calculus with the same expressive power as the ADL fragment of PDDL. However, it was not proven that the identified restrictions define a maximum subset. For some of the restrictions we will show that they are indeed necessary. For the others we will examine to what extent we can ease them without loosing the same expressiveness as ADL.

Malte Helmert

Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning

Heuristic search is a leading approach to domain-independent planning. For cost-optimal planning, however, existing admissible heuristics are generally too weak to effectively guide the search. Pattern database heuristics (PDBs), which are based on abstractions of the search space, are currently one of the most promising approaches to developing better admissible heuristics.

The informedness of PDB heuristics depends crucially on the selection of appropriate abstractions (patterns). Although PDBs have been applied to many search problems, including planning, there are not many insights into how to select good patterns, even manually. What constitutes a good pattern depends on the problem domain, making the task even more difficult for domain-independent planning, where the process needs to be completely automatic and general.

We present a novel way of constructing good patterns automatically from the specification of planning problem instances. We demonstrate that this allows a domain-independent planner to solve planning problems optimally in some very challenging domains, including a STRIPS formulation of the Sokoban puzzle.

Jan-Georg Smaus

Using Predicate Abstraction to Generate Heuristic Functions in UPPAAL

We focus on checking safety properties in networks of extended timed automata, with the well-known UPPAAL system. We show how to use predicate abstraction, in the sense used in model checking, to generate search guidance, in the sense used in Artificial Intelligence (AI). This contributes another family of heuristic functions to the growing body of work on directed model checking. The overall methodology follows the pattern database approach from AI: the abstract state space is exhaustively built in a preprocess, and used as a lookup table during search. While typically pattern databases use rather primitive abstractions ignoring some of the relevant symbols, we use predicate abstraction, dividing the state space into equivalence classes with respect to a list of logical expressions (predicates). We empirically explore the behavior of the resulting family of heuristics, in a meaningful set of benchmarks. In particular, while several challenges remain open, we show that one can easily obtain heuristic functions that are competitive with the state-of-the-art in directed model checking. We will also discuss some recent work on generating predicates by an abstraction refinement loop.