Combining Learning and Programming for High-Performance Robot Controllers

Alexandra Kirsch, Michael Beetz

Combining Learning and Programming for High-Performance Robot Controllers

Alexandra Kirsch and Michael Beetz

Lehrstuhl für Informatik 9, Technische Universität München

Abstract. The implementation of high-performance robot controllers for complex control tasks such as playing autonomous robot soccer is tedious, error-prone, and a never ending programming task. In this paper we propose programmers to write autonomous controllers that optimize and automatically adapt themselves to changing circumstances of task execution using explicit perception, dynamics and action models. To this end we develop ROLL (Robot Learning Language), a control language allowing for model-based robot programming. ROLL provides language constructs for specifying executable code pieces of how to learn and update these models. We are currently using ROLL's mechanisms for implementing a rational reconstruction of our soccer robot controllers.

1 Introduction

The implementation of high performance robot controllers for complex control tasks such as playing autonomous robot soccer is tedious, error-prone, and a never ending programming task. Optimizing robot behavior requires parameter tweaking, situation specific behavior modifications, and proper synchronizations between concurrent threads of control. The mutual interactions between parameter settings, synchronizations, and situation-specific hooks imply that even tiny changes to the program may have large and unpredictable effects on the robot behavior.

Even worse, not only changes to the program cause drastic behavior changes, but even changes to the environment such as switching from one soccer field to another, changing lighting conditions, or simply playing against another team might cause the same controller to produce very different behavior. In one case we played on a softer carpet which caused the robot to sink deeper and the ball to get stuck between the guide rail for dribbling and the floor. As a result, the dribbling skill and the action selection had to be substantially modified. The robots had to be equipped with mechanisms to detect situations where the ball is stuck in order to avoid overheating the controller board. Or, in another case the playing strategy had to be extended substantially when we played against a team with bigger robots. The bigger robots occluded landmarks on the field and the ball. As a consequence the robots needed much more sophistication in situations where they couldn't localize themselves and the ball [2].

In order to enable programmers to implement controllers that can adapt themselves quickly to such changing circumstances we propose to specify the code pieces requiring adaptation or optimization based on experience explicitly as executable learning problems. For this purpose we develop a language ROLL (Robot Learning Language) that

integrates learning mechanisms into the existing robot control and plan language RPL [4]. Using ROLL programmers can specify robot controllers that represent, acquire, and reason about models of their control routines. In this paper we focus on the acquisition of appropriate models by automatic learning. Besides models we also learn routines that execute the robot's desired actions using the automatically acquired models.

With this approach the learning process is executable and can be performed automatically. Therefore the models and parts of the control program can be relearned without human interaction whenever the environment changes substantially. But the integration of learning and programming is not only a convenience to the developper, it also enhances the learning performance. Robot learning is a complex task that requires the adjustment of parameters like the experiences and the learning bias. On the one hand, these parameters can be found empirically in a convenient way, as they are represented explicitly and the learning process is performed automatically. On the other hand, these parameters could be acquired by a meta-learning process.

Still, an explicit representation of the learning process does not suffice to make a robot learning task reproducible, because there is a huge uncertainty in the training experiences. To get the experiences under control we explore the use of datamining techniques. The field of datamining has yielded strong mechanisms for producing clean, understandable data. We found that the learning performance can be enhanced significantly by applying data mining techniques.

In order to underline our arguments, figure 1(a) shows a code fragment of a simple control program for an autonomous soccer robot. The parts in italics denote the functions or routines that require an adaptation to the environment. We see that in this case all the routines needed to fulfill the goals could be learned. Also, the choice of the next goal could be made by a learned decision tree depending on the opponent team.

```
repeat
  cond
    opponent-attack? → defend-own-goal
    near-ball?       → score-goal
    supporting-mate? → support-team-mate
    else             → annoy-opponent
```

(a) Simple robot controller.

```
with-models
  action-model, perception-model, ...
seq
  tagged learning-phase
    to-be-learned ← extract-non-operational-routines()
    for-every problem in to-be-learned
      learn(problem)
  tagged execution-phase
    repeat
      current-goal ← choose-goal(current-situation)
      try-all
        wait-for tasks-changed?
        achieve current-goal
```

(b) Controller with learned models.

Fig. 1. Example controller for an autonomous soccer robot with and without learned models.

In contrast, figure 1(b) shows a control program that uses explicit, learned models. The first step in the execution is to learn all the routines that have not been learned yet.

After that the robot starts its normal course of action. Here the choice of the next goal is performed by a learned function and also the execution routines are learned.

In the rest of the paper we glimpse at the main concepts of our learning language ROLL. Then we will briefly go into how we tested parts of this language and how we plan to evaluate it in the future. We will conclude with section 4.

2 Learning Mechanisms

Figure 2 shows the parts of a learning agent. Every aspect of a learning problem is represented explicitly within ROLL.

The *performance element* realizes the mapping from percepts into the actions that should be performed next. The control procedures therein might not yet be executable or optimized. These are the procedures we want to learn.

The *critic* is best thought of as a learning task specific abstract sensor that transforms raw sensor data into information relevant for the learning element. To do so the critic monitors the collection of experiences and abstracts them into a feature representation that facilitates learning. The experiences are stored in a *database*, which allows us to employ datamining mechanisms for data cleaning.

The *learning element* uses experiences made by the robot in order to learn the routine for the given control task. It can choose a learning algorithm from a library of so-called *learning systems*.

The *problem generator* generates goals that are achieved by the performance element in order to gather useful experiences for a given learning problem. The problems are generated according to a probability distribution as given in the learning problem specification.

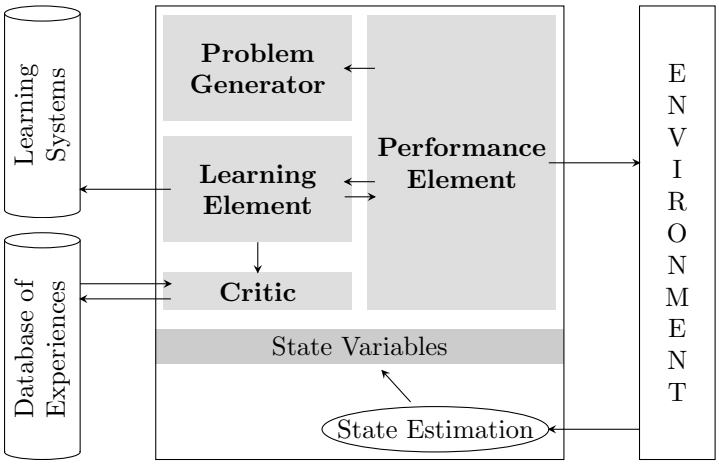
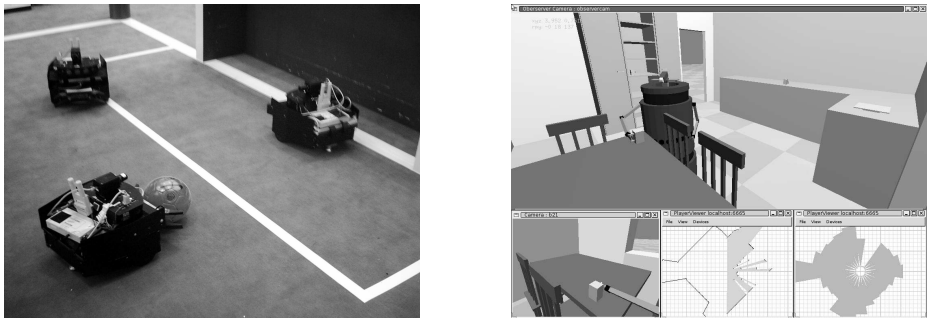


Fig. 2. Parts of a learning agent after Russell and Norvig.

3 Progress and Further Work

We are currently using ROLL's mechanisms for implementing a rational reconstruction of our autonomous soccer robot controllers and for the control of an autonomous household robot (figure 3). The salient features of the code are that it is self-optimizing and adaptive.



(a) Autonomous soccer robots.

(b) Realistic household simulation.

Fig. 3. Evaluation testbeds for ROLL: robotic soccer and a simulated household robot.

We used the learning mechanisms described in section 2 for learning navigation tasks in the domain of robot soccer. The learned routines were successfully applied in the RoboCup world championship 2004 in Lisbon [3,1]. The model-based paradigms were also employed on that occasion, as well as in the domain of the household robot.

The next step in our research is to combine the underlying model-based language with the learning concepts, so that the scenario in figure 1(b) can be implemented as shown. We will then be able to perform extensive testing on how the overall performance changes when the learning parameters are changed. After that we intend to extend the learning constructs for other forms of learning like reinforcement learning.

4 Conclusion

We have given a very brief overview over our robot learning language ROLL, which combines learning and programming in a synergetic way. By first learning appropriate models (perception, actions, dynamics), we then proceed to learning control routines and decision functions.

The learning process is described explicitly in the control program and is executed automatically by the controller. This makes it repeatable and reproducible. Besides, the parameterization for a learning problem on a special robot can be carried over to different robot platforms and similar learning problems.

References

1. M. Beetz, A. Kirsch, and A. Müller: RPL-LEARN: Extending an autonomous robot control language to perform experience-based learning. 3rd International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS), 2004.
2. M. Beetz, T. Schmitt, R. Hanek, et al.: The AGILO robot soccer team experience-based learning and probabilistic reasoning in autonomous robot control. Autonomous Robots, 17(1):55–77, 2004.
3. M. Beetz, F. Stulp, A. Kirsch, et al.: Autonomous robot controllers capable of acquiring repertoires of complex skills. RoboCup International Symposium 2003, Padova, July 2003.
4. D. McDermott: A Reactive Plan Language. Research Report YALEU/DCS/RR-864, Yale University, 1991.