Application of Qualitative Reasoning to Robotic Soccer

Gordon Fraser and Gerald Steinbauer and Franz Wotawa[1]
Technische Universität Graz (TUG)
Institute for Software Technology
Inffeldgasse 16b/II, A-8010 Graz, Austria
{fraser,steinbauer,wotawa}@ist.tu-graz.ac.at

Abstract

In this paper we focus on qualitative reasoning aspects of knowledge representation for robotics and argue why they are appropriate for building robust control systems. We describe how to extract qualitative knowledge from a numerical world model, how to use this model for computing plans, and how to execute these plans. Robotic soccer is used as a real-world example domain. Problems might arise with regard to reactivity, unreliable sensor data might lead to stability issues and unintended behaviour. These drawbacks need to be treated. However, in general using a qualitative representation to do reasoning with offers several advantages such as robustness and efficiency.

Introduction

The development of mobile and truly autonomous robots requires a robust control system that takes the sensed world, the internal state, and their knowledge base into account to compute actions in order to achieve a given goal or purpose. The focus on robustness for control systems is very important. Here, robustness of a control system stands for invariance of the computed output to small deviations between input data. Obviously, we further require a control system to be active, i.e., to come up with actions from time to time. Without this second requirement a control system that does nothing would be the best choice for a robust system. The term small in the definition of robust control systems heavily depends on the situation and therefore on the underlying knowledge. For example, in soccer it is rather unimportant whether a ball is 1 or 2 meters away from a player providing there is no opponent which is closer to the ball.

We can distinguish between two different techniques for robotic control systems. Reactive systems [1] are system comprising a set of tiny control actions that are activated or deactivated directly based on the sensed inputs. The control loop of reactive systems is very small and reactivity is high. Hence, such systems are very promising in situations where a fast response is required. Classical AI control systems comprise sensing devices, a mapping from the sensor data to the abstract representation, a planner, and an action executor which maps the abstract representation back to the actuators. The control loop of AI control systems is quite large and therefore reaction time usually is low. The advantage of classical AI systems is that they provide the knowledge in an accessible format and that they usually consider more informations which improves robustness and clarity of decision making. A good view of the two approaches is to consider reactive systems as an implementation of reflexive actions whereas classical AI systems are implementing the more strategic planning approach.

In this paper we focus on qualitative reasoning aspects of knowledge representation for robotics and argue why they are appropriate for building robust control systems. The underlying robotic architecture we have in mind is a combination of the reactive system and the classical AI approach, also known as *hybrid systems*. The lower level tasks are implemented using techniques from reactive systems whereas the high level tasks like developing and executing strategies follow classical approaches. In the paper we describe how to extract knowledge from the world model, how to use this model for computing plans, and how to execute these plans. The first part of the control loop, i.e., the extraction of a world model from the sensors and the mapping from the quantitative world model to the qualitative world model is depicted in Figure 1. In this figure the qualitative representation is given as a set of ground literals which describe relationships between objects and qualitative descriptions of spatial information. This qualitative information together with assigned tasks, e.g., the goalie's task in robotic soccer, is used by a planner to compute a new plan. The plan is then executed. During plan execution the high level actions are mapped to low level actions like navigation via path planning. This mapping is implemented via a procedure which is assigned to each high level action.

The paper is organized as follows. In the next section we explain why a qualitative representation is very appropriate for high-level control systems. We discuss the mapping from the qualitative world model to the quantitative world model and show how some problems can be avoided (to some extent). Afterwards, we introduce our planning framework which is followed by an example. Finally, we discuss related research and conclude the paper.

Qualitative Representation of the Quantitative World

There are several techniques available for generating a model of reality which is based on observations, i.e., the available sensor input. These techniques include the use

[1] authors are listed in alphabetical order

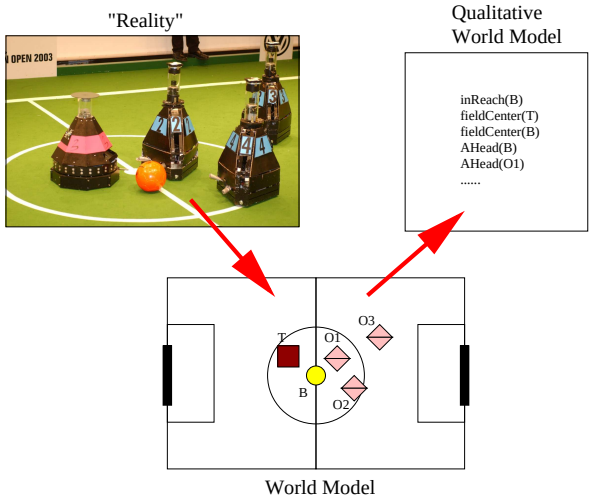


Figure 1: From the real world to its qualitative representation

of Kalman filters [2], Markov methods [3] and particle filter [4]. All of them have in common that the resulting model comprises object locations with a corresponding distribution function that indicates the degree of certainty in the location. From all possible object locations the best one, usually in terms of the one with the highest probability or the one with the highest degree of certainty, is chosen. Obviously, the resulting object location is a quantitative description of the world.

The drawback of using only the quantitative values for making a decision in a specific situation at a given point in time becomes obvious when considering the soccer situations of Figure 2. All of the three soccer situations are different with respect to their corresponding quantitative model. However, from a high-level view situations (a) and (b) become equivalent because in both the ball B is in reach of player T and the way to the goal is not blocked by the opponent goalie O. Note that it seems that (a) and (b) are different because of the different distances to the goal but for the player's T view it makes (almost) no difference. Situation (c) is a different situation as the way to the goal is blocked.

In principle a purely quantitative model has to consider an infinite number of situations in order to plan new tasks. This is obviously not possible and usually some sort of analysis has to performed which classifies situations. However, this would again result in a qualitative description of the world in terms of situation classes. Hence, from a static view the qualitative description of the world state allows to represent an infinite number of possible situations by using a finite alphabet. The remaining questions to be answered regard the alphabet in order to describe a language for representing different situations and resulting actions, and how quantitative information can be compiled into its qualitative representation.

The qualitative representation language for a specific situation comprises ground predicates P which express basically facts about the perceived state of the world. These facts represent the factual knowledge of the world an agent has. This could be knowledge about the visibility of objects or relationships between objects like $reachable(A, B)$ stating that the objects A and B are close together.

The interpretation I of a n-ary predicate $p \in P$ depends on the world model M, i.e., the quantitative description of the world, and can be formalized as follows (where X^n represents the grounded argument vector of size n):

$$I(p(X^n), M) = \begin{cases} true & \textbf{if } COND_p(X^n, M) = true \\ false & \textbf{otherwise} \end{cases}$$

The function $COND_p$ has to be implemented for every predicate p and uses the model M and the predicate's argument vector as the input. For example, the predicate $inReach(X, M)$ may be implemented as follows:

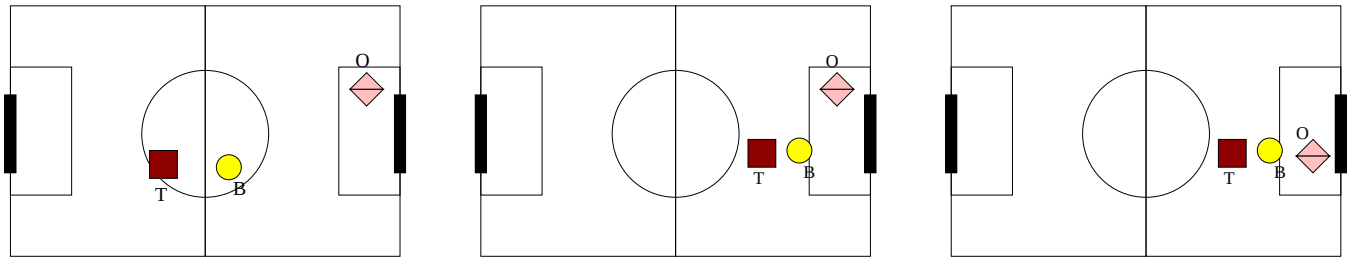
```
COND_inReach(X,M)
    return dist(pos(X,M),pos(self,M)) < 1
```

In `COND_inReach` the function `dist` returns the distance of two quantitative positions, `pos` returns the position of an object in the current world model M, and `self` returns the agent itself. All functions are built-in functions of the agent's control system. The behavior of `COND_inReach` is that it returns true if the object is within a 1 meter distance from the agent.

In practice the definition of I gives rise to problems. One problem is due to the sharp boundary where a predicate changes its truth value. For example, there is no clear distinction whether to say `inReach` must be true if an object is in a 1 or 1.1 meter distance from the agent. However, because we are dealing with predicates this sharp boundary cannot be avoided. Another more severe problem is entailed by reality. In a real environment the world model is not only imprecise but also unstable. For example, small deviations in lighting conditions can cause large changes in the perceived world which is represented by the world model. As a matter of fact these deviations in the world model may cause a predicate to be true at a time and false at the immediately following point in time although there are no changes in the real world which are of interest for the agent.

In order to retain stability we have to reformulate the definition of the interpretation function for predicates. The basic idea behind the reformulation is that once a predicate evaluates to true, it should remain true unless a significant large change in the world model occurs. This behavior is well known in electrical engineering as *hysteresis function*. Systems whose behavior follow a hysteresis function can be seen as systems with a state that remembers a previously taken decision. Hence, the extension of the interpretation function I has to consider the logical state of the predicate. The new interpretation function I^H can be defined as follows:

$$I^H(p(X^n), M, s) = \begin{cases} true & \textbf{if } COND_p(X^n, M, s) = true \\ false & \textbf{otherwise} \end{cases}$$



(a) (b) (c)

Figure 2: 3 different situations in robotic soccer

In this definition the variable s represents the current truth value of p. Obviously, the function $COND_p$ also has to be redefined in order to implement a hysteresis function. For example, the *inReach* predicate may have the following `COND_inReach` function assigned to.

```
COND_inReach(X,M,s)
    if s then
      return dist(pos(X,M),pos(self,M)) < 1.3
    else
      return dist(pos(X,M),pos(self,M)) < 1.0
```

With this hysteresis the predicate *inReach* remains true unless the distance between the agent and an object X is greater than 1.3 meters. Hence, the hysteresis introduces safety with respect to world model changes that are caused by exogenous events like changes in illumination and sporadic misinterpretation of features. An alternative approach to modeling uncertain data is the use of probabilistic methods. However, this causes the disadvantage of having to deal with probabilities in the reasoning process.

Figure 3 depicts an example for the evaluation of the predicate *inReach* using the prior hysteresis function. In Situation (a) the ball B lies within 1 m distance to player T. In Situation (b) the ball B did not move but the world-model reports a wrong position for the ball B'. In Situation (c) the ball B has moved. In (d) the evaluation sequence of the predicate *inReach* is shown.

The introduction of hysteresis functions is always a trade-off between the stability of a predicate and its reactivity. Consider a hysteresis function for a predicate that always returns false. This function is optimal with regard to stability in terms of the number of undesired changes of the truth value (jitter) but aside of this it is useless for qualitative modeling.

From the engineering perspective the following actions have to performed for developing a qualitative representation as explained in this section:

1. Identify a set of predicates. The predicates are used to represent spatial knowledge and relationships between objects and agents.

2. Write a function $COND_p$ for every predicate p which implements a hysteresis function if necessary.

3. Adapt the hysteresis function in order to gain stability.

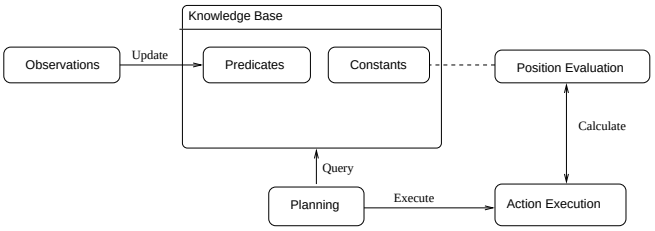


Figure 4: Abstracting numerical data to symbolic data, performing qualitative reasoning on this symbolic data, and grounding the resulting actions to create numerical motion commands.

In the following section the planning system including the actuator part of the control system is explained.

Planning Framework using Qualitative Reasoning

Robot controllers receive information about the world in a numeric form from sensors, and they must supply precise numeric information in order to position and control actuators. As they act in the physical world, also for data processing some form of spatial representation is necessary. Vision systems, for example, need an explicit spatial representation since they operate on a two dimensional projection of the physical subject. Robots using multiple sensors to gather data need a common spatial representation that can be used to fuse data to a unified world model. However, this representation does not necessarily need to be the same used for reasoning. Explicit data can be mapped to a qualitative, symbolic representation usable for planning, and vice versa for plan execution. As depicted in Figure 4, observations made by the robot are used to calculate the truth value of predicates that describe the state of the observed world. Predicates and constants contained in a qualitative knowledge base are used by a planning module. The resulting plan contains actions that are not directly executable by the robot's actuators, first the qualitative symbols need to be mapped to an explicit representation again.

Planning tasks are particularly appropriate for the application of qualitative techniques. Several advantages are gained by performing planning with a qualitative representation.

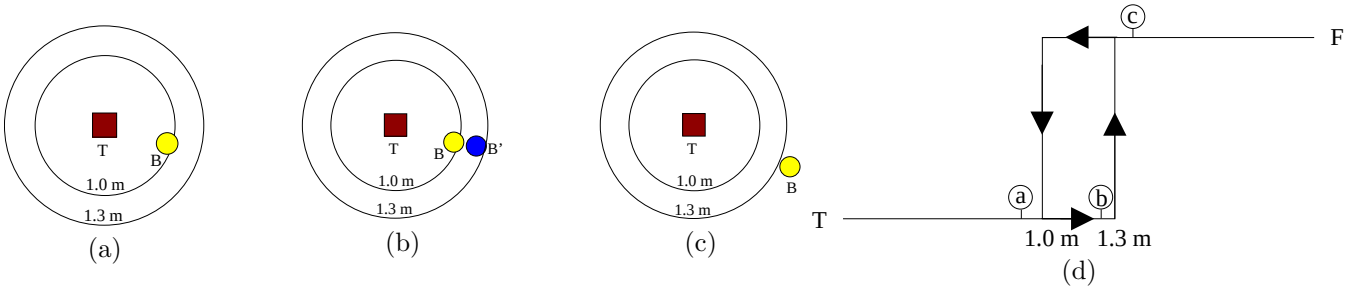


Figure 3: Evaluation of the predicate *inReach* using a hysteresis function.

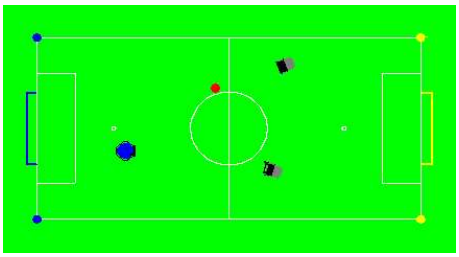


Figure 5: Example robotic soccer situation. Image is a screen-shot of Simsrv [5], a Middle-Size league simulator. Connected agents are running the implementation of the described planning framework.

- Graceful degradation is achieved, as even in situations where the robot encounters errors in its data it should be able to continue operating.
- High-level robot programming is greatly simplified for human operators.
- The size of the search space for the planning problem is cut down significantly.
- Qualitative models are able to cope with uncertain and incomplete knowledge.
- A qualitative model equals an infinite number of numeric models

Example

This section shows a simplified scenario in robotic soccer. The soccer robot in Figure 5 depicted by the hexagon is the planning agent, there are two opponent robots to the right. Simplified, the agent has the following model of the current world state:

$(Left(OwnGoal) \wedge Ball(Ball) \wedge Ahead(OpponentGoal) \wedge Attacker() \wedge Ahead(Ball) \wedge Goal(OwnGoal) \wedge NextTo(OwnGoal) \wedge Left(OpponentGoal) \wedge Goal(OpponentGoal) \wedge Left(Ball) \wedge Behind(OwnGoal) \wedge Heading(OpponentGoal))$

Knowledge Representation Constants used in this context are the main objects involved in playing soccer:

Ball

OpponentGoal

OwnGoal

$Ball(x)$ and $Goal(x)$ are static predicates describing objects. $Left(x)$, $Right(x)$, $Ahead(x)$, $Behind(x)$ are relative disposition predicates that are used to choose appropriate goals.

BallSensed indicates whether the ball is placed in front of the robot's ball control device, and is set by a sensor.

InReach(x) Indicates that object x is within a certain distance to the robot, i.e., its focus of attention.

LinedUp(x,y) Indicates whether two objects are in line with the agent.

Reachable(x) is defined as $\forall x(InReach(x) \wedge LinedUp(Ball, x) \wedge BallSensed() \leftrightarrow Reachable(x))$

IsAt(x,y) Indicates whether the object x is at the position y.

Distinct positions are labeled and can be used for qualitative reasoning as well. For example, strategic positions such as a good defense position for a robotic soccer robot can be used just like any other symbol. A description for such a position could look like so:
$IsAt(Self, DefensePosition) \leftrightarrow Blocked(OwnGoal, Ball) \wedge InReach(OwnGoal)$.
In this sentence $Self$ stands for the planning agent. Such positions need to be somehow mapped back to explicit spatial information in order to execute actions appropriately. This mapping is implemented via the potential field method proposed in [6].

Actions This section lists actions that are relevant for finding a solution to the given example. Action descriptions are given in STRIPS-like form [7], i.e., they consist of a precondition that has to be fulfilled in order to execute the action, and an effect that will be fulfilled after executing the action.

Aim:

Parameter: $target$

Precondition: $\neg BallSensed() \wedge InReach(Ball)$

Effect: $LinedUp(Ball, target)$

DribbleTowards:

Parameter: $target$

Precondition: $BallSensed() \wedge LinedUp(Ball, target)$

Effect: $Reachable(Ball, target)$

GoTowards:

Parameter: $target$

Precondition: $\neg BallSensed()$

Effect: $InReach(target)$

GrabBall:

Precondition: $LinedUp(Ball, OpponentGoal) \wedge InReach(Ball)$

Effect: $BallSensed()$

Kick:

Parameter: $target$

Precondition: $Reachable(target)$

Effect: $IsAt(Ball, target)$

Planning Algorithm The application of qualitative reasoning greatly simplifies the domain description. Lower level tasks such as path-planning or precise action execution are performed by a separate action execution module that does not need to do symbolic reasoning. It has to map symbolic target descriptions to real coordinates, however. The greatly reduced average plan length and the small number of objects to be considered as action parameters cuts down the size of the search space considerably. In fact, a simple state space regression planner [8] was used in the first implementation:

Iterative deepening search is used, the stack nodes consist of an open condition and a plan. At each iteration, all actions that fulfill the current condition are prepended to the current plan and added to the open search nodes.

The opponent's actions do not need to be considered by the planning algorithm. Instead, plan invariants [9] are used to monitor plan execution. These invariants observe changes in the environment, e.g. opponent's counter actions, and cause re-planning if necessary.

Example Plan Considering the initial state description given at the beginning of this section and the domain theory stated in a previous section, the only thing that is necessary to describe a classical planning problem (consisting of initial state, domain theory and goal) is a goal description. As the robot in the example is a soccer playing agent, a worthwhile goal would be to score a goal. This could for example be expressed as: $IsAt(Ball, OpponentGoal)$

After a few milliseconds of thinking, the agent comes up with the following plan:

1. GoTowards: Ball
2. Aim: OpponentGoal
3. GrabBall
4. DribbleTowards: OpponentGoal
5. Kick: OpponentGoal

Related Work and Conclusion

In [10] [11] hybrid systems for controlling robots of a RoboCup MSL Team were presented. Both use *Golog* for the representation of the qualitative model and the derivation of plans. Furthermore, they use the qualitative model and decision trees to evaluate the most appropriate action in a certain situation. This action is used if no plan is available in that situation. The used action might not be the best but keeps the robot reactive even the planning take some time. However, the problems arising from noise and jitter in the quantitative model were not accounted in those approaches.

There is other work in the MBR/DX community for compiling quantitative models into qualitative models, e.g. AQUA[12]. In contrast to our work, [12] requires the complete knowledge of the quantitative model. We do not require such a quantitative model because we only map quantitative observations to qualitative observations.

We have shown how qualitative knowledge-representation can be applied to real-world robotic agents. Classical AI-Planning is suited very well to perform reasoning with this knowledge. We have pointed out the major problems this approach has and proposed solutions. This approach is clearly qualitative where quantitative information is mapped to their symbolic equivalent. Problems arising with this mapping, e.g., jitter phenomena due to unreliable sensor data, can be avoided using hysteresis functions.

We believe our approach offers a lot of potential. However, the problems that were pointed out in this paper show that further research is necessary. In particular empirical evaluation is required in order to compare the mapping from quantitative to qualitative information with and without hysteresis. Another issue is due to the cognitive aspects of intelligent agents. It might be of interest not to formalize the actions but to formalize knowledge about the physical world and its rules. With such a knowledge representation it should be possible to derive actions directly. Hence the physical world model can be seen as meta knowledge. For example, passing a ball obeys the laws of physics.

References

[1] Rodney A. Brooks. Intelligence without reason. In John Myopoulos and Ray Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[2] P. S. Maybeck. The Kalman filter, An introduction to concepts. In *Autonomous Robot Vehicles*, pages 194–204, 1990.

[3] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[4] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2), 2001.

[5] A. Kleiner and T. Buchheim. A Plugin-Based Architecture for Simulation in the F2000 League. In *Proc. of the International RoboCup Symposium*, 2003.

[6] T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J.-S. Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski. Cs Freiburg 2001. In *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*. Springer, 2002.

[7] Richard. E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1972.

[8] Daniel S. Weld. An Introduction to Least Commitment Planning. *AI Magazine*, 15(4):27–61, 1994.

[9] Gordon Fraser and Franz Wotawa. Cooperative Planning and Plan Execution in Partially Observable Dynamic Domains. In *Proc. of the International RoboCup Symposium*, 2004.

[10] F. Dylla, A. Ferrein, and G. Lakemeyer. Acting and Deliberating using Golog in Robotic Soccer - A Hybrid Architecture. In *The Third International Workshop on Cognitive Robotics*. AAAI, 2003.

[11] Vasco Pires, Miguel Arroz, and Luis Custodio. Logic Based Hybrid Decision System for a Multirobot Team. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems*, 2004.

[12] M. Sachenbacher and P. Struss. AQUA: A Framework for Automated Qualitative Abstraction. In *Working Papers of the 15th International Workshop on Qualitative Reasoning (QR-01)*, 2001.