Using Model-Based Diagnosis to Build Hypotheses about Spatial Environments

A Response to a Technical Challenge

Oliver Obst*

Universität Koblenz-Landau, AI Research Group
Universitätsstr. 1, D–56070 Koblenz, Germany
fruit@uni-koblenz.de

Abstract. We present a method to build a hypothesis on the condition of the environment in which a robotic multi-agent team moves. Initially the robots have a default assumption about the conditions of the floor and on how moving under these condition works. For certain parts of the environment however, the default assumption may be wrong and moving around does not work in the expected way. Now the robotic team builds a hypothesis on the conditions of the yet unvisited part of the environment in a way similar to computing a diagnosis for electrical circuits. Resources can be saved by avoiding areas that possibly also contain obstacles.

1 Introduction

In RoboCup Simulation League, the simulated robots are situated in a two-dimensional world where objects move according to some fixed rules, which are known to the players. Usually, during a regular tournament, these rules are not going to change. Additionally to the regular tournament during RoboCup 2001 there was an evaluation round where parts of the physics of the simulation was changed before the matches started, without that programmers knew of this in advance. The change in the physics had the effect that dashing on the upper half of the field resulted in only half of normal speed for all the players. Some of the teams did manage these matches better than others, but to the best of our knowledge none of the teams could come up with a diagnosis of what happened on the field, though this was immediately visible for human spectators.

In this paper, we present a solution to the problem how a team of robotic agents can come up with a hypothesis of what might globally be wrong with the environment in similar situations. Hypotheses should entail the currently observed behavior and provide some kind of "forecast" for those areas that no other member of the team visited so far. As time proceeds the hypothesis will be refined to match the actual situation more closely. With a hypothesis about the condition of the environment, single robots can try to avoid areas that potentially contain obstacles, take the shortest way out of these areas, or instead enter these areas to (dis-)prove the hypothesis dependent on the strategy of the team.

* This research is supported by the grant *Fu 263/6-1* from the German research foundation *DFG*.

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 518–525, 2004.

2 From Problems to Hypotheses

What we essentially want the robots to do is related to the approach of model-based diagnosis: abnormalities have to be identified and a kind of explanation for these abnormalities should be generated (see for instance [4]). Though we identify some differences between doing model-based diagnosis for devices and building hypotheses for robots in the first part of this section, we can apply the same procedure to compute the result in our approach.

2.1 Differences between Model-Based Diagnosis for Devices and Building Hypotheses for Robots

Though at first sight both problems seem to be very similar, there are differences with respect to some general points:

Devices vs. Space. Model-based diagnosis is usually used for physical devices consisting of several components. The "device" we are interested in in our approach is space, i.e. the floor on which the robots move. To use a procedure like model-based diagnosis, we artificially have to introduce components like smaller areas which make up space.

Diagnosis vs. Hypothesis. Even if we introduce kinds of devices in both approaches there is a difference with respect to result that should be computed: By taking a description of the system and a description of the behavior of the system, model-based diagnosis is usually used to explain the observed behavior by providing a diagnosis which states possible faulty components. For our robots, we observe the – possibly faulty – behavior of some of the components (parts of an area) and by this observation, we want to build a hypothesis on the behavior of the whole system (the complete area).

2.2 Identifying Abnormalities

The only way our robots can deduce the presence of an obstacle at their current location is by moving forward and using position estimation or odometric information, that is robots can execute a kind of `move` operation and recognize if it succeeded or if it failed. A description of the (usual) behavior of the `move` operation, an observation of the behavior of the `move` operator and some knowledge concerning the accuracy of these both are necessary to decide if a robot actually has a problem or not.

3 Representation of the Environment

The assumptions made in the previous section enable us to do two things: Firstly, robots can clearly identify a part of the field as being defective if `move` does not work in a particular situation. Secondly, robots can exchange information on abnormal tiles by sending logical facts to each other. These informations need only to be sent in unexpected situations. By doing so, robots exchange and collect information about tiles on the field

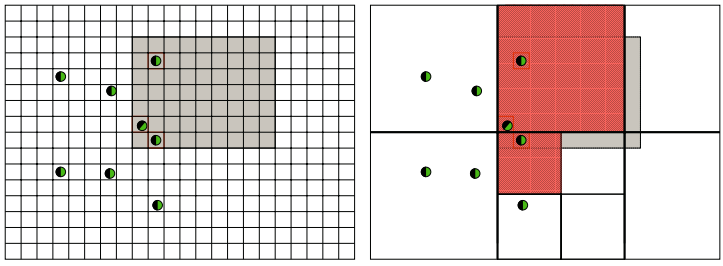

Fig. 1. Left: Area partitioned into atomic tiles with robotic team. On the gray tiles, robots cannot move as fast. **Right:** Created hypothesis from this situation.

they already visited. To get an idea of how the yet unvisited tiles could be like, we set up rules that relate tiles of different levels to each other. The basic assumption here is that if there is an unexpected change in the environment, this change usually concerns areas larger than the size of our atomic tiles. If there is a tile containing an obstacle, its unvisited neighbors will probably also contain obstacles. In the next subsection we are going to explain how we want to relate tiles of different levels to each other. We will assume that the area is partitioned into squares, but other ways of partitioning are possible. In the second part of this section we are going to describe the logical rules we are using for the relation between tiles.

3.1 Hierarchical Layering of Tiles

To build a hypothesis on defective tiles, we use a hierarchical layering of tiles inspired by quad trees [6]. In the case of squared tiles, we use 2×2 atomic squares (level 0) to make up a higher-level square (level 1), 2×2 squares of level 1 to make up a square of level 2 and so forth. The last layer in this hierarchy covers the whole area accessible for the robots. Like the atomic tiles, each of the higher-level tiles gets a symbolic label so that it can be identified uniquely.

The idea behind building a hierarchical representation using tiles is the following: Initially, all robots have the default assumption that `move` succeeds in the whole area. If, for a tile of a certain level, it is known that it covers only unknown (smaller) tiles and at least one tile where `move` succeeds, our hypothesis should be that `move` will work for the complete tile. If the tile covers only unknown smaller tiles and at least one tile where `move` fails, our hypothesis should be that `move` will fail for the complete tile. Tiles containing both kind of smaller tiles (and possibly unknown tiles) have to be split up into tiles of the next lower level. For an example hypothesis see the graphics on the right in Fig. 1.

3.2 Logical Representation

Each tile in our representation can be identified by its level and its coordinates, so the tiles are the "components" of the field. Logically, a tile is an atom $a_{l,x,y}$ with level l, where x and y denote the coordinate of the respective tile in that level. The non-atomic

tiles contain lower level tiles. These connections including the resulting assumptions on the state of respective tiles can be described using logical rules.

In model-based diagnosis, a model of a device is used to compute the expected output given a particular input. Discrepancies between expected and the actual output are used to detect faults in the system. In our setting, the attempt to move on a certain position can be regarded as "input" to an atomic tile. Thus, we denote the fact that the robot tries to move on location $a_{0,i,j}$ by $\text{move}(a_{0,i,j}, i)$.

The result of this attempt can be regarded as "output" of an atomic tile, and so we denote the success of the operation `move` by $\text{move}(a_{0,i,j}, o)$, while a failure of this operation is denoted by $\neg\text{move}(a_{0,i,j}, o)$.

There can be different kinds of faults in a system like this: in our approach, we are aiming at faulty tiles, the components in the system. Possible are also faults in the actuators or in the sensors measuring the output of the actuators; however for now we ignore them.

Definition 1. *An atomic tile $a_{0,i,j}$ is called defective (or abnormal), if it is known that the `move` operation fails on $a_{0,i,j}$ (written: $\text{ab}(a_{0,i,j})$). An atomic tile $a_{0,i,j}$ is called normal if it is known that the `move` operation succeeds on $a_{0,i,j}$ (written: $\neg ab(a_{0,i,j})$).*

In our approach we use two definitions concerning the connections between atomic and higher level tiles: Definition 2 deals with implications for a higher level tile from knowing that the `move` operation succeeds on an atomic tile, while Definition 3 describes implications for higher level tiles from knowing that the `move` operation failed. A third possibility for an atomic tile is that no robot visited the tile so far, so that there is no knowledge on the behavior of the `move` operation on the respective tile.

Definition 2. *A tile of level l ($l > 0$) containing a normal atomic tile is not abnormal.*

This relation between atomic tiles and higher level tiles can be expressed used logical formulæ. The number of formulæ for each tile is dependent on the level of the tile and the used topology. In the case of square tiles the branching factor is 4, i.e. a tile of level one contains four level 0 tiles, while a tile of level two contains sixteen level 0 tiles.

Example 1. In the case of square tiles consisting of four lower level tiles the rules for Definition 2 look like this:

$$\begin{aligned}
\neg ab(a_{1,0,0}) &\leftarrow \text{move}(a_{0,0,0}, o) \wedge \text{move}(a_{0,0,0}, i).\\
\neg ab(a_{1,0,0}) &\leftarrow \text{move}(a_{0,0,1}, o) \wedge \text{move}(a_{0,0,1}, i).\\
\ldots\ldots\ldots &\leftarrow \ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
\neg ab(a_{2,0,0}) &\leftarrow \text{move}(a_{0,0,0}, o) \wedge \text{move}(a_{0,0,0}, i).\\
\ldots\ldots\ldots &\leftarrow \ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned}$$

Definition 3. *(At least) one of the tiles containing an abnormal atomic tile is abnormal.*

Example 2. In the case of square tiles consisting of four lower level tiles the rules for Definition 3 look like this:

$$
\begin{array}{rcl}
ab(a_{0,0,0}) \vee ab(a_{1,0,0}) \vee ab(a_{2,0,0}) & \leftarrow & \neg \mathrm{move}(a_{0,0,0}, o) \wedge \mathrm{move}(a_{0,0,0}, i). \\
ab(a_{0,0,1}) \vee ab(a_{1,0,0}) \vee ab(a_{2,0,0}) & \leftarrow & \neg \mathrm{move}(a_{0,0,1}, o) \wedge \mathrm{move}(a_{0,0,1}, i). \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \leftarrow & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
ab(a_{0,3,3}) \vee ab(a_{1,1,1}) \vee ab(a_{2,0,0}) & \leftarrow & \neg \mathrm{move}(a_{0,3,3}, o) \wedge \mathrm{move}(a_{0,3,3}, i). \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \leftarrow & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{array}
$$

The number of formulæ of this kind necessary is equal to the number of atomic tiles for the complete area. The right hand side of each formula denotes that `move` failed on an atomic tile, while on the left hand side we have a disjunction of all tiles that contain the respective atomic tile – including the atomic tile itself. The length of the right hand side of these formulæ is logarithmic with respect to the number of atomic tiles.

4 Building a Hypothesis

Once the set of clauses as described in the last chapter is available, collected knowledge from moving around and from communication and a theorem prover can be used to solve the problem of building the actual hypothesis. We need to compute models of the given set of clauses, where the extension of the *ab*-literal is minimal. That means the theorem prover used should find solutions for the given clauses so that there exists no solution containing a subset of true *ab*-literals. A procedure to compute minimal diagnosis with a theorem prover can be found in [1], the theorem prover `NIHIL`[1] we have been using to compute hypotheses is based on this procedure. We briefly explain some basics for this procedure in the following subsection; in principle any other procedure for computing minimal models could be used.

4.1 Model-Based Diagnosis with Hyper Tableaux

Usually, for a diagnosis we need a system description (*SD*), a set of components of the system (*COMP*), and an observation (*OBS*). As stated earlier, a component c can be abnormal ($ab(c)$), or it can behave normal ($\neg ab(c)$). According to Reiter [9], a diagnosis is defined as follows:

Definition 4 (Reiter 87). *A Diagnosis of* $(SD, COMP, OBS)$ *is a set* $\Delta \subseteq COMP$*, such that* $SD \cup OBS \cup \{ab(c) | c \in \Delta\} \cup \{\neg ab(c) | c \in COMP - \Delta\}$ *is consistent.* Δ *is called a* Minimal Diagnosis*, iff it is the minimal set (wrt.* $\subseteq$*) with this property.*

As we shall see later, the set of observations is simply a set of logical facts. A basic ingredient to the diagnosis method mentioned above are hyper tableaux [2]. Basically, a hyper tableau is a finite ordered tree T where all nodes, besides the root node, contain literals from a finite set $\mathcal{S}$ of ground clauses. A branch b in T is called regular if each literal in b occurs at most once, otherwise it is called irregular. A tree T is regular iff all its branches are regular. Branches containing containing contradicting literals are labeled as closed, and as open otherwise. A tableau is closed if each of its branches is closed, otherwise it is open. Computing minimal diagnoses is possible by finishing all open branches and collecting the *ab*-literals from open branches.

[1] NIHIL = New Implementation of Hyper In Lisp. Thanks to Peter Baumgartner for providing helpful information on the calculus and on using his system.

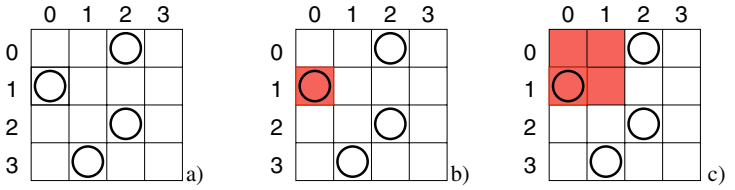

Fig. 2. Example with one robot on a defective tile (0,1) and two possible hypotheses.

4.2 Selecting a Hypothesis

Even if the computed diagnosis are minimal, for a given situation there could be several minimal ones. Consider a robot put on the center of a square finding the atomic tile on its place defective. In each level of the hierarchy of tiles there is one tile surrounding the atomic tile that could be a hypothesis for being a defective tile. The least conservative hypothesis covers only atomic tiles which are known to be defective. The most conservative hypothesis expands to the largest tiles that contain at least one abnormal, but no normal atomic tile.

Example 3. Consider the four robots from Fig. 2. The facts about their current locations are like this:

$$\begin{aligned} &\text{move}(a_{0,0,1}, i) \leftarrow \top. && \bot \leftarrow \text{move}(a_{0,0,1}, o).\\ &\text{move}(a_{0,2,0}, i) \leftarrow \top. && \text{move}(a_{0,2,0}, o) \leftarrow \top.\\ &\text{move}(a_{0,1,3}, i) \leftarrow \top. && \text{move}(a_{0,1,3}, o) \leftarrow \top.\\ &\text{move}(a_{0,2,2}, i) \leftarrow \top. && \text{move}(a_{0,2,2}, o) \leftarrow \top. \end{aligned}$$

So it is known that tile $a_{0,0,1}$ is defective, whereas tiles $a_{0,2,0}$, $a_{0,1,3}$ and $a_{0,2,2}$ are known to be working. There is nothing known about the other tiles. The rules describing the relation of tiles between the different levels are selected as in Example 1 and Example 2. One diagnosis as shown in Fig. 2 b) ($ab(a_{0,0,1})$) will be computed, and the other one as shown in Fig. 2 c) ($ab(a_{1,0,0})$). To select the most conservative hypothesis, one has to choose the diagnosis with the least number of faults. If there is more than one diagnosis with this property, the one containing tiles $a_{l,i,j}$ with the largest levels l is the most conservative one.

4.3 Why Logic?

Looking at the hypothesis and the way it gets computed the question arises why one should do it this way instead of using the quad trees directly. There are different answers to this question.

Firstly, we have chosen to use a theorem prover because it was available. Creating the rules to represent the environment was done automatically by a simple program, so the programming effort to solve the problem was very small. But there are other advantages of this approach:

Flexibility. The approach using a theorem prover is flexible with respect to the selected topology, because the representation is independent of the computation. A robot moving in a building consisting of several different rooms can select a topology dependent on the room it is in.

Robustness. The use of logic for communicating facts about the environment can add robustness to the negotiation process among different team members. We assume that robots communicate an observation only if it contradicts their current hypothesis about the environment. By receiving known information robots can possibly still improve their current hypothesis, because it can be deduced that a previous message must have been lost.

Use of Additional Knowledge. By using additional knowledge about potential obstacles, the selection of hypotheses can be controlled. This is possible because the theorem prover computes all minimal diagnosis, from which we can select an appropriate one. By choosing the hypothesis containing the tiles with the largest levels we select the most conservative one. Dependent on knowledge or assumptions about the environment we can also prefer hypothesis with obstacles of a certain size, if applicable.

However, a disadvantage of our approach is that with larger areas and increasing number of tiles calculation of hypothesis becomes slow. This is due to the fact that the number of formulæ increases quadratic with the number of tiles.

5 Related Work

As mentioned earlier, model-based diagnosis [9, 4] is usually concerned with computing an explanation for a certain behavior of technical devices. Logical descriptions of the device and of the components of the device are used to predict the expected behavior of the device, given a particular input. The diagnostic task is to identify possible faulty components by comparing the actual with the predicted output. The *ab*-literal is used to denote faulty components. Approaches as described in [1] take the system description and the observation of the behavior as a set of clauses to compute models so that the extension of the *ab*-literal is minimal. Besides the reasoning strategies the assumptions made for the reasoning process are a matter of concern, see for example [5]. For an overview about symbolic diagnosis in general see [7].

As in our approach to specify and implement a team of agents [8], the authors in [3] use a logical approach to specify the knowledge of an agent. The agent architecture ALIAS is introduced, where agents are equipped with hypothetical reasoning capabilities. Hypothesis are raised by an agent and it is checked if the hypothesis complies with the other agents knowledge, contrary to our approach where the knowledge is collected prior to raising hypotheses.

6 Conclusions and Future Work

In this paper we showed a way to build hypotheses about the environment of robotic teams with a theorem prover. A hypothesis about unknown areas can be useful to guide

the selection of actions, so that resources can possibly be saved. The logical description of the environment can be generated automatically for different topologies. Because the method to actually compute the hypothesis is separated from the description of the environment, topologies could be switched at run time. Should the time of computation take too long due to a large number of atomic tiles the approach can be used to compare different kinds of topologies and to be reimplemented in a more efficient fashion later on. In our approach we looked at changes in the environment only, incorporating sensor and actuator faults into our model is left to future work. Other points for future work are methods to recover from message loss, and using methods to acquire knowledge from teammates without communication. For large areas with a low density of facts a logical description with variables could be tried.

References

1. Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, and Wolfgang Nejdl. Semantically Guided Theorem Proving for Diagnosis Applications. In M. E. Pollack, editor, *15th International Joint Conference on Artificial Intelligence (IJCAI 97)*, pages 460–465, Nagoya, 1997. Morgan Kaufmann.
2. Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper Tableaux. In *Proc. JELIA 96*, number 1126 in Lecture Notes in Artificial Intelligence. European Workshop on Logic in AI, Springer, 1996.
3. Anna Ciampolini, Evelina Lamma, Paola Mello, Cesare Stefanelli, and Paolo Torroni. An implementation for abductive logic agents. In Evelina Lamma and Paola Mello, editors, *AI*IA 99: Advances in Artificial Intelligence*, LNAI 1792, pages 61–71. Springer, Berlin, 2000.
4. Luca Console and Pietro Torasso. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence*, 7(3):133–141, 1991.
5. Dieter Fensel and Richard Benjamins. Assumptions in model-based diagnosis. In B.R. Gaines and M.A. Musen, editors, *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, KAW'96*, pages 5/1–5/18, Calgary, 1996. Department of Computer Science, University of Calgary, SRDG Publications.
6. Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.
7. Peter J.F. Lucas. Symbolic diagnosis and its formalisation. *The Knowledge Engineering Review*, 12(2):109–146, 1997.
8. Jan Murray, Oliver Obst, and Frieder Stolzenburg. Towards a logical approach for soccer agents engineering. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, number 2019 in Lecture Notes in Artificial Intelligence, pages 199–208. Springer, Berlin, Heidelberg, New York, 2001.
9. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987.