RoboCup Logistics League TDP
Graz Robust and Intelligent Production System
GRIPS

Sarah Haas, Dino Keskic, Clemens Mühlbacher,
Gerald Steinbauer, Thomas Ulz, Marco Wallner

February 29, 2016

Abstract

This paper describes the approaches we use to tackle the challenge arising in the RoboCup Logistics League 2016. As platform the Robotino 2 with small modifications and additional hardware (laser scanner, camera, external notebook) is used. The software architecture consists of a scheduler/planner (multi-robot auctioning approach) on top of a procedural reasoning engine (Open PRS) and the robot operating system (ROS) in the lowest layer. The vision system for light pattern detection is based on a histogram of gradients (HOG) detector and a feed-forward neural network (FFNN). To improve the robustness of the system, observers monitor the low-level and report abnormal behavior and it's likely reasons to the upper layers using a diagnosis system to allow to react to the issue.

1 Introduction

The RoboCup Logistics League aims to serve as a testbed for methods dealing with smart production. This is achieved by simulating a smart factory where different products should be manufactured using production machines. The main goal is to transport the resources, which are necessary to create an ordered product, between the different machines. This is done by using of fleets of autonomous robots. Thus this setting simulates the logistics problems in future production systems. Therefore it is a perfect testbed for new planning/scheduling algorithms and methods for controlling fleets of robots.

Due to the interesting challenges posed by this league the GRIPS team was founded. The GRIPS team was initiated as part of the course "Construction of Mobile Robots" hold at Graz University of Technology. To improve our knowledge about known problems and to be able to learn from other teams already participating in this league, a part of the team also attended the "RoboCup Logistics Winter School" held at the the RWTH Aachen University and organized by the winner of the last years, Carologistics. On the basis of the experience there and the results gained during the lecture, the proposed hardware and software architecture was developed. In this paper, the hardware

modifications on the Robotino 2 as well as the used additional hardware are described in Section 2. The algorithms and software architecture used is described in Section 3. The next chapter briefly discusses the overall mission strategy. In Section 5 we present our development process. The next section builds a bridge to ongoing research at the institute. In Section 7 we draw a brief conclusion on the proposed approach.

2 Hardware

As a base we are using three Robotino 2 [1] by Festo (see Fig. 1b). They are equipped with a customized construction to achieve the right height for the mounted gripper to be able to grab the products as they are lying on the conveyors. Furthermore the Robotinos are equipped with an external PC to gain adequately computational power to run the proposed software system. For navigation there is a laser scanner (Sick TIM551) mounted as well as a camera to detect the QR-tags. To achieve the needed precision at the conveyor, an additional camera and if necessary an additional laser scanner is mounted at the level of the gripper. To detect the light pattern one additional camera is mounted on the robot. Finally one external PC is used to coordinate the robots and reserve the usage of the production machines.

3 Software

Following the idea of a three-layer architecture [2] the software is structured as follows: scheduler/planner, mid-level control and low-level. We introduced a strict communication scheme where only adjacent layers communicate with each other to achieve an increasing abstraction of the real world from layer to layer. Lower layers provide functionality to the higher layers (see Fig. 1a).

The high-level is responsible for coordinating the robots to achieve the common goal. Thus the high-level is in charge of distributing tasks each robot has to perform and to ensure that two robots don't use the same resource at the same time.

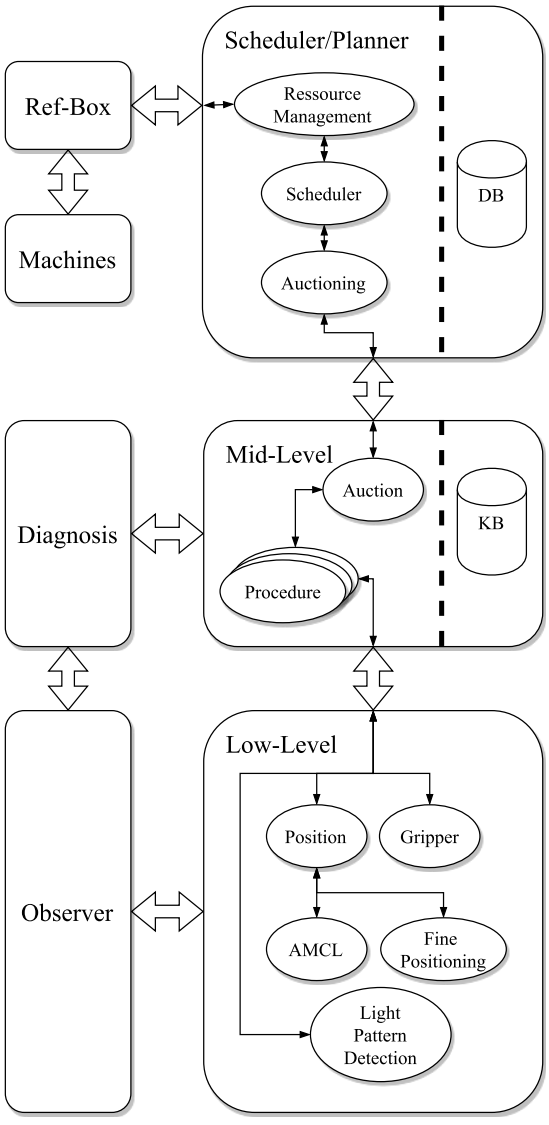
The mid-level is responsible to control one robot. Thus the mid-level plans which actions have to be executed to finish the tasks given by the high level. Additionally the mid-level is responsible to recover from faults in the task execution as far as it is possible for the robot to resolve the problem by itself.

The low-level is responsible to execute primitive actions. These action range from detecting the light-pattern to moving between way-points. This level is the most reactive one and is the only level which performs near real-time activities.

Each level is described in more detail in the remaining subsections of this section.

3.1 Scheduler/Planner

The scheduler/planner is on top of the three layer structure and has therefore the most abstract view. It is the only instance communicating with the referee box and is therefore responsible to keep track of the orders and to forward the robot status arriving



(a) Overview of the software architecture.

(b) Adapted Robotino 2.

Figure 1: Used software architecture and hardware setup.

from the lower layers. As there is only one scheduler/planner for all the robots, the resource management is done here as well.

The resource manager keeps track of the current state of the production machines and their usage. Thus only one robot is allowed to use the production machine at the same time. This avoids incnsistent configurations of the production machines.

The scheduler/planner maintains a database containing a pool of active tasks, the game state, the state of the robots and the state of the machines. This is used to distribute the tasks (i.e. atomic production steps to be made to fabricate a product) to the robots in order to maximize the points. To distribute the tasks in an efficient way, they are sold at an auction.

3.1.1 Auctioning of Tasks

When a new production step has to be performed, the scheduler offers it to all the connected robots. They can respond to this with their approximated costs for this job. The tenderer with the lowest cost will win the bid and get the instruction to perform this step. The other robots wait for the next auction to compete in. If all robots are currently busy (this is known by the scheduler/planner), the tasks to execute next are queued and auctioned if one or more robot finishes it's current task. A similar approach can be found in [3].

The prioritizing of the tasks is not trivial since each task has a time window in which it has to be performed, i.e. the scheduler has to assert that the robot does not finish too soon or too late. Additionally the potential number of points achievable for each task have to be taken in consideration to perform optimal too (e.g. a task to deliver a final product has a significantly higher priority than a task to fetch a base because delivering a product achieves more points than fetching a base). To gain the maximum flexibility, a sold task can be canceled by the scheduler/planner if a task already needed too long and the product will not be finished in time.

In order to find the best task to auction next, different approaches are currently under evaluation. These approaches differ in the expected performance concerning computational costs and maximizing the number of points. The current used approach advertise those tasks which are related to the product with the earliest dead line (earliest deadline first, EDF). This approach is lightweight and scores moderately. Another approach which is currently under evaluation uses a constraint solver [4]. Here the points to gain by finishing a task, the dependencies of a task, an estimation of the task duration and the dependencies within the tasks in order to find the schedule which maximizes the point score is modeled. This approach is computational expensive but we expect better scoring results.

3.1.2 Tasks

Production orders received from the referee box in the production phase are split up in a set of atomic tasks (i.e. further decomposition is not possible or makes no sense) one robot can perform (see Fig. 2). They are published during the auction as protobuf messages. These messages contain information about the tasks, such as an unique ID of the task, the type of the task, time of issuing and the deadline as well as the involved

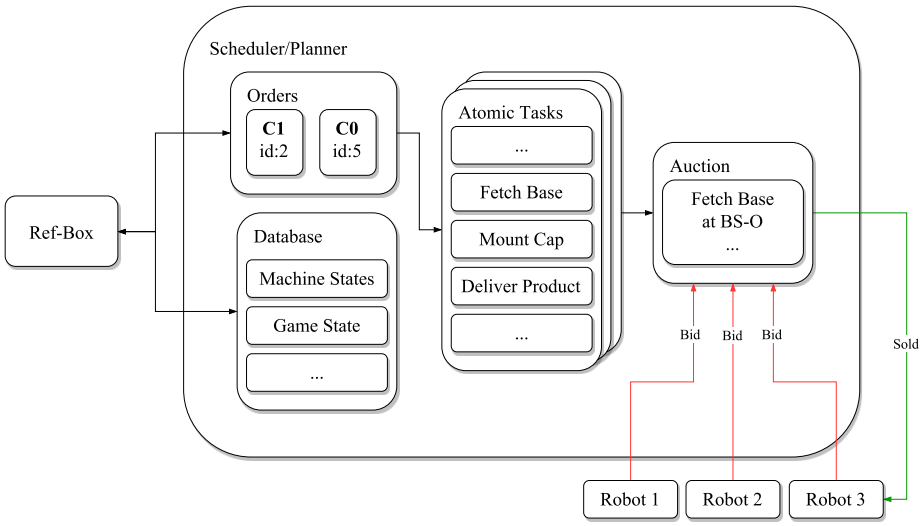


Figure 2: Decomposition of orders into atomic tasks and the auctioning approach.

machines. Please note the task are always sold if there is a robot willing to perform the task unless it viaolates a deadline. Through the auctioning process each robot can bet on an offered tasks. Furthermore the server assigns the tasks to the robot. Finally the robots issue feedback after a task has finished.

3.2 Mid-Level Control

The mid-level control is mainly based on the open Procedural Reasoning System (Open PRS) [5]. Open PRS implements the belief-desire-intention (BDI, [6]) software model and allows to draw operational plans (OPs) using a graphical user interface. Within a BDI system the agent keeps track of the current state of the world and the goals which should be achieved. To achieve a goal, the agent selects one OP. Through this paradigm the robot can achieve a task depending on the environment and the robot status. The integrated reasoning engine also allows parallel execution of these OPs, e.g. used in the exploration phase to simultaneously explore and look for unseen QR-tags.

As a production step is auctioned by a robot, the corresponding OP is called to achieve this goal. An example of an OP to close the gripper can be seen in Figure 3. This example should illustrate some functionalities and the syntax. This OP is initiated by posting the invocation part `(achieve (gripobject))` using the message passing (MP) interface. The plan is only applicable if the gripper is open at the moment, i.e. `((test (gripper open))`. After successful execution of this plan, `(gripper-holds-object)` is written in the database. On the edges of the graph, subgoals can be posted and (evaluable) predicates can be tested. E.g. the posting of the goal `(gripper closed)` is directly forwarded to the lower level as it is an atomic part of the execution chain.

Additionally error detection and error handling of low-level functions is done in the

mid-level. As far as possible it is determined what went wrong and what is the most effective way to recover from this error. In order to do so the mid-level comprises a knowledge base containing the state of the robot and its environment to find an efficient recovery if there exists one. It is important to notice that any failure detected by the diagnosis system (see Section 3.4 for details) is also reported to the mid-level and is stored in the knowledge base. Thus the mid-level also tries to recover from faults detected through the diagnosis system.

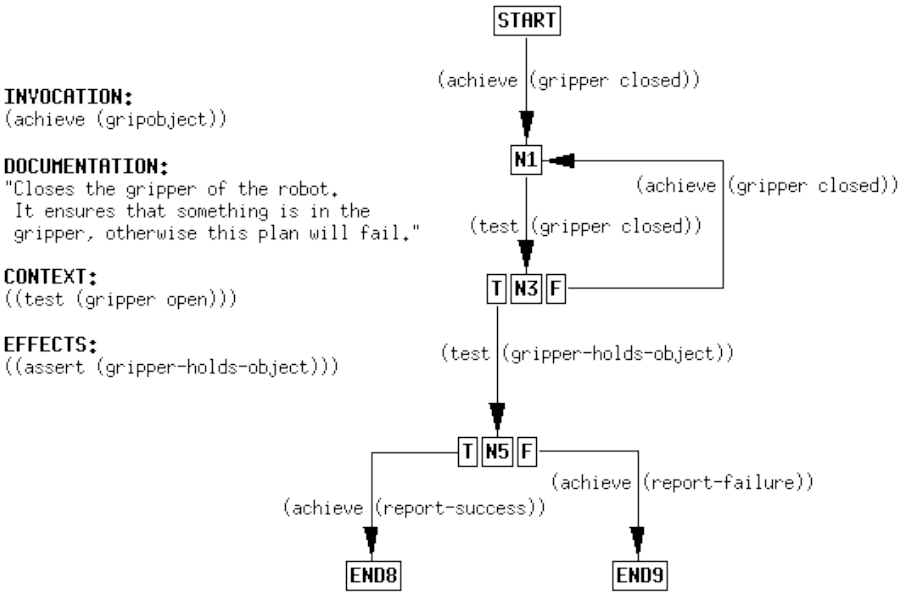


Figure 3: Sample OPRS plan to grab some object with the gripper.

3.3 Low-Level

The lowest layer is mainly based on the open robot operating system (ROS [7]). Here, basic atomic actions are advertised to the layer above as a ROS action server. Different services are advertised here, e.g. open/close the gripper, locally navigate to a machine, align in front of a machine and so on. Performing these actions, all the error detection and as far as possible error recovery is made. In the following part of this subsection we will briefly review the most important parts of the low-level functions.

3.3.1 Light Pattern Detection

The detection of the light patterns during the exploration phase is done in two steps. First, the region of the image containing the lights is extracted using a histograms of oriented gradients (HOG) detector (a similar approach for human detection, see [8]). Doing this, the regions of interest can be extracted as it can be seen in Figure 4. Having

(a) All lights off. (b) Green light on. (c) Orange light on.

Figure 4: Examples for extracted signal lights using HOG detector.

this regions, the second step, i.e. the decision which pattern is presented, can be reduced to a standard classification problem using a feed-forward neural network (NN)e lights in the captured image, the classification practically works every single time. The used features are the RGB values of all the pixels. For this, the images are sampled up or down (as needed) to a fixed size (size of the lights in an usual distance for detection, e.g. 64×192 pixels) using bilinear interpolation. With this approach, the neural network consists of 12288 input neurons, eight hidden units and seven output neurons (one out of n coding for the classes).

3.3.2 Way-Point Navigation

In order to retrieve or deliver objects, the robot moves between defined way-points. These way-points are stored in the mid-level as abstract identifiers e.g. the output of the base-station as `BS-O`. In order to move to a given way-point, the low-level uses a table to look up the real world coordinates for the abstract identifier and afterwards plan to move to this way-point. To move to the given coordinates, we use the move_base package of ROS [9]. This package comprises a global planner which uses the map to find a path between the current robot position and the goal position. In order to avoid the different production machines, these machines are added in the navigation map. Furthermore after finding a global plan a local planner is used to move along the path by considering detected objects.

3.3.3 Conveyor Alignment

During the production phase the robot needs multiple times to deliver and retrieve products from the conveyor. This is done through a way-point navigation near to the end of the conveyor. Afterwards the QR-Code and the laser scanner is used to approach the machine and to align the robot close to the conveyor belt. As a very precise alignment is needed in order to retrieve something from the conveyor, the information provided by the camera is used too. For this, the conveyor is detected through an additional camera and a HOG detector (see Figure 5 for an example result). The detector is trained in such a way that the bounding rect is centered above the conveyor entrance, i.e. the region of interest is bounded at the bottom with the two guide rails. This information can be

used in addition to achieve exact alignment. For further improvement of the precision, the use of a second laser scanner below the gripper is evaluated. This additional data can be used to detect the entrance of the conveyor too.

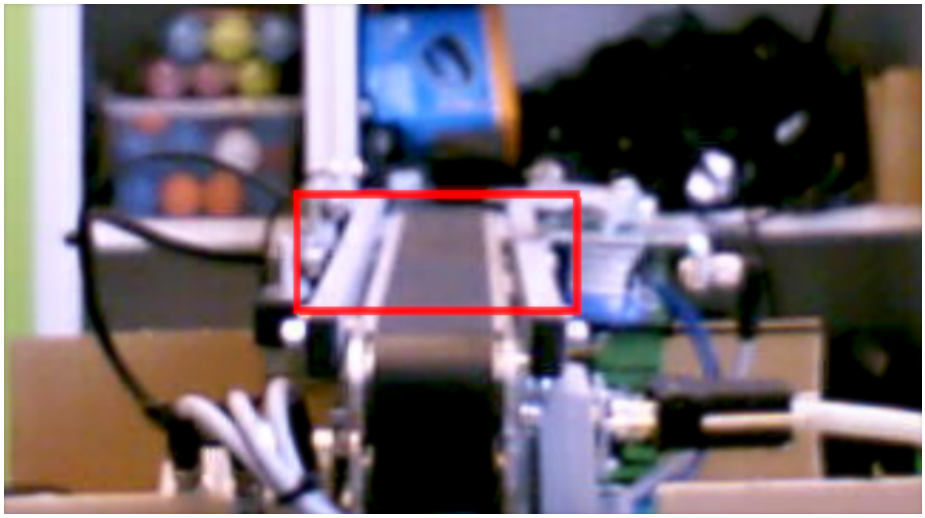


Figure 5: Detected conveyor entrance used for fine-alignment.

3.4 Diagnosis System

In order to guarantee a certain degree of dependability, an online diagnosis system is used. The diagnosis system is a more efficient re-implementation of the method described in [10]. The diagnosis system uses a set of observers to monitor different properties of the system e.g. the frequency of the communication between two different low-level modules. These observations are checked if they are within the nominal system behavior. If the observation yields a discrepancy, a diagnosis is calculated. This is done using a diagnosis engine based on PyMDB [11] which calculates a consistency based diagnosis [12]. Thus the result of the diagnosis is a set of low-level modules which, if assigned to be faulty, would explain the undesired observations. The information of the possible faulty components is reported to the mid-layer in order to allow an repair mechanism to be triggered.

3.5 Communication

The communication between the scheduler/planner and the mid-level control is done in a similar way as the communication with the referee box has to be done using serialized messages generated with the open protocol buffer data format (protobuf) developed by Google Inc. Also the internal communication in the mid-layer is done using such serialized streams as Open PRS has some limitations for the communication. The mid-level control and the low level communicate via ROS action server, i.e. the low-level

offers actions which can be triggered by the mid-layer. Doing this it is possible to get the current status of the action during execution as well as a return status as the execution is finished or failed.

4 Mission Strategy

The game can be split up into two main phases, exploration and production. Therefore also the strategies can be described separately.

4.1 Exploration

In the exploration phase each robot will be in charge of exploring one column of our side of the field. Therefore the robot moves on the field and searches for unseen QR-tags. As one is found, the robot drives in front of the corresponding machine. Here, the signal light can be extracted using the HOG detector and classified with the neural network. The collected information is then sent to the highest layer. The scheduler/planner gathers all reports from the robots with additional information about the certainty of these observations. Having multiple observations with given probabilities, a safe report can be sent to the referee box to avoid negative points due to wrong reports. Exploring only one half of the field is sufficient due to it's symmetry property.

4.2 Production

The three-layer architecture described in Section 3 of our approach comes into account here. The scheduler splits up the orders into atomic tasks to build a desired product and holds them in a task pool. The tasks are auctioned to the robots ordered by their priority to achieve a maximum number of points. The scheduler/planner keeps the overview to achieve a global maximum. Such atomic tasks are e.g. providing a cap, mounting a cap, discarding the base and so on. All the robots bid for the tasks (if they are idle) with their current local cost, i.e. the robot doing the job most efficiently wins the bid. As a robot needs a modular production system (MPS), it tries to get a mutual lock for this at the scheduler/planner. There, a global locking table is stored to avoid congestion and multiple configurations of machines.

5 Development

In order to design the system robust we use during our development an continuous integration server which executes beside builds also unit and integration tests. These integration tests are executed with the help of the Gazebo simulation which is provided by the BBUnits and the Carologistics team [13, 14]. Through this continuous testing software faults and integration problems can be found more easily. To encourage other teams to follow the idea of continuous integration, we plan to release the software to perform these tests after it has shown to be mature.

6 Influence Through Current Research

Within the focus of the current research at the institute hosting the team is the usage of a model driven approach to develop robots. The idea is to use models as a central part of the developing process to automatically test and diagnose the running system. This approach has shown to be applicable for an industrial use case. Thus we hope that this approach also results in a robust system for the logistic league.

Furthermore current research is done in order to design an agent architecture which allows an easy integration of a model driven approach into a robotic system. This architecture should allow a robotic system to be more dependable and thus run for long periods of time. We expect that the result of this research is of special interest for the logistics league as it should reassemble a smart production.

Finally we have investigated different high-level approach to control the robot fleet together with the Carologistics team [15]. The high-level approach evaluated was based on YAGI (Yet Another Golog Interpret), which is an implementation based on the ideas of the logic-based agent control language GOLOG. YAGI allows the usage of imperative as well as declarative parts. Thus programming and planning can be balanced between ease of use and performance.

7 Conclusion

2016 will be the first year of competing in the RoboCup Logstics league. We use a central planning and scheduling instance which distributes the tasks to perform through an auctioning system. The robots use a BDI system in order to execute the tasks in an reliable and reactive manner. Additionally the robot will be observed during the runtime in order to detect faults and allow a fast recovery. Through the research performed at the institute hosting the team, we expect to give the league new ideas allowing to create more dependable systems.

References

[1] Karras, U., Pensky, D., Rojas, O.: Mobile robotics in education and research of logistics. In: IROS 2011–Workshop on Metrics and Methodologies for Autonomous Robot Teams in Logistics. (2011)

[2] Gat, E., et al.: On three-layer architectures. Artificial intelligence and mobile robots **195** (1998) 210

[3] Nunes, E., Gini, M.: Multi-robot auctions for allocation of tasks with temporal constraints. In: Twenty-Ninth AAAI Conference on Artificial Intelligence. (2015)

[4] Yokoo, M., Hirayama, K.: Algorithms for distributed constraint satisfaction: A review. Autonomous Agents and Multi-Agent Systems **3**(2) (2000) 185–207

[5] Ingrand, F.F., Chatila, R., Alami, R., Robert, F.: Prs: A high level supervision and control language for autonomous mobile robots. In: Robotics and Automation,

1996. Proceedings., 1996 IEEE International Conference on. Volume 1., IEEE (1996) 43–49

[6] Georgeff, M., Pell, B., Pollack, M., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. In: Intelligent Agents V: Agents Theories, Architectures, and Languages. Springer (1998) 1–10

[7] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA workshop on open source software. Volume 3. (2009) 5

[8] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 1., IEEE (2005) 886–893

[9] Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The office marathon: Robust navigation in an indoor office environment. In: International Conference on Robotics and Automation. (2010)

[10] Zaman, S., Steinbauer, G., Maurer, J., Lepej, P., Uran, S.: An integrated model-based diagnosis and repair architecture for ros-based robot systems. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE (2013) 482–489

[11] Quaritsch, T., Pill, I.: Pymbd: A library of mbd algorithms and a light-weight evaluation platform. Proceedings of International Workshop on Priciples of Diagnosis (DX) (2014)

[12] Reiter, R.: A theory of diagnosis from first principles. Artificial intelligence **32**(1) (1987) 57–95

[13] Zwilling, F., Niemueller, T., Lakemeyer, G.: Simulation for the robocup logistics league with real-world environment agency and multi-level abstraction. In: RoboCup 2014: Robot World Cup XVIII. Springer (2014) 220–232

[14] Niemueller, T., Reuter, S., Ewert, D., Ferrein, A., Jeschke, S., Lakemeyer, G.: Decisive factors for the success of the carologistics robocup team in the robocup logistics league 2014. In: RoboCup 2014: Robot World Cup XVIII. Springer (2014) 155–167

[15] Ferrein, A., Maier, C., Mühlbacher, C., Niemueller, T., Steinbauer, G., Vassos, S.: Controlling logistics robots with the action-based language yagi. In: IROS Workshop on Taks Planning for Intelligent Robots in Service and Manufacturing. (2015)