Simulation Competitions on Domestic Robots

Jianmin Ji, Zhiqiang Sui, Guoqiang Jin, Jiongkun Xie, and Xiaoping Chen*

Multi-Agent Systems Lab, School of Computer Science and Technology
University of Science and Technology of China, 230026, Hefei, China
{jizheng,zqsui,abxeeled,devilxjk}@mail.ustc.edu.cn, xpchen@ustc.edu.cn

Abstract. This paper reports a series of simulation competitions on domestic robots. All of these five competitions were based on a simulation platform focused on evaluating high-level functions of a domestic robot, including task planning and dialogue understanding. The object of holding these competitions is to promote research and development of service robots while avoiding limitations imposed by hardware of real robots. We also analyze the results and performances of participating teams since the competition was first held in 2009, showing that more and more terms are participating and they are performing better and better.

Keywords: RoboCup@Home, domestic robots, simulation platform, task planning, dialogue understanding

1 Introduction

Researchers from Artificial Intelligence (AI), Robotics and related areas have shown increasing interest in developing intelligent service robots [1, 3, 8, 12]. One of the most promising applications for a service robot is to provide services for untrained and non-technical users at home. Then, as a part of RoboCup, RoboCup@Home league [13] was held to develop service robots for future personal domestic applications and the RoboCup@Home competition is held each year since 2006. In the competition, a number of standard tests are used to evaluate robots' functions and performance in a realistic non-standardized home environment setting. These tests focus on functions which are essential for domestic applications including human-robot interaction, task planning, navigation, mapping, vision, object recognition, object manipulation, system integration and so on. However, due to the limitations of hardware and complexity of robotics techniques like vision, navigation, etc, it is not easy to test the different realizations of high-level cognitive functions of a real robot frequently or to develop a real robot to participate in competitions such as RoboCup@Home. In this paper, we report an effort against these limitations.

Five competitions have been held so far. All of them are based on the same simulation platform, though it has been upgraded several times. The platform

* Corresponding author.

is intended to evaluate the performance of a robot on task planning and dialogue understanding. A typical application scenario of a robot is to extract and understand users' requirements and information from dialogue through natural language interface, then resolve corresponding tasks and compute a plan of motions and sub-tasks to meet these requirements. Clearly, these two functions are indispensable for domestic applications, in addition to robots' hardware and underlying technologies in robotics. The platform simulates the low-level functions of an ordinary domestic robot and the features of common home environments related to the tests, by sending to each competing program a list of testing problems expressed in some verbal languages. The competing programs are required to try to solve all the testing problems, ie, to understand each problem and generate a plan for it within a given time limit. The competing programs are evaluated in terms of the performance of the plans they generate.

The first competition was held on December 2009 with 4 teams, while in 2011 two competitions were held with 12 teams and more challenging testing problems. In this paper, we analyze the results of all five competitions. It shows that more and more teams are participating and they are performing better and better. It also indicates that the platform can be used to compare different approaches for task planning and dialogue understanding of a domestic robot.

The rest of the paper is organized as follows. Section 2 presents the simulation platform. Section 3 and 4 report the results of competitions and compare the different approaches employed by the participating teams. Further discussions and conclusions are given in Section 6 and Section 7, respectively.

2 A Simulation Platform for Task Planning and Dialogue Understanding of Domestic Robots

Task planning and dialogue understanding play essential roles in the development of a domestic robot's high-level functions. In principle, these functions can be realized by various approaches. Then it is extremely interesting to compare these approaches in solving the same problems that involve these functions under the same conditions.

For this purpose, we developed a software platform for testing the relevant high-level functions of competing programs which may be developed by different approaches. The platform simulates the low-level functions of an ordinary domestic robot which could automatically move to a specific place, has an arm with a gripper to manipulate small objects and a plate to handle an object each time. The platform also simulates the features of common home environments related to the tests, including the location of an object, whether an object is portable, etc. Human-robot dialogue is simulated in a simplified way, by sending to each competing program a list of testing problems expressed in some verbal languages.

The competing programs are required to try to solve all the testing problems, ie, to understand each problem and generate a plan for it within a given time

limit, 5 seconds. The competing programs are evaluated in terms of the performance of the plans they generate. Obviously, the simulated tests on the software platform are much simpler than what can be done with a real robot. But we get much more experimental data from the competitions on the platform, which in turn make the comparisons between different approaches possible.

Now we show some details.

The Primitive Actions of the Simulated Robot A set of primitive actions are pre-defined in the platform. They keep fixed for all the testing problems. Following the AI convention, each primitive action is specified by its preconditions and effects. In the original version of the platform, there are five primitive actions, listed below.

- $move(X)$: The robot moves and arrives at location X.
- $pickup(A)$: The robot picks up object A.
- $putdown(A)$: The robot puts down object A.
- $toplate(A)$: The robot puts object A in its plate.
- $fromplate(A)$: The robot picks object A up from its plate.

Note that there is a plate on the robot, so that the robot can carry two objects, one in the plate and the other in its gripper. The definitions of these actions are also specified in PDDL statements[1].

The Testing Problems Each testing problem is specified by a scenario description and a task description. The scenario description specifies the initial state of the home environment, which simulates the robot's perception since a real robot perceives its environment state through its sensors. The task description consists of one or more goals, constraints, and other additional information which the user tells the robot when he/she requests a specific service.

A scenario description provides the information of the types of the objects appeared in the scenario, their locations and other attributes. It also provides the current state of the robot. A scenario description is stored as a file in the following form. The objects and agents existing in the scenario are assigned a unique positive integer, denoted as num. In particular, number 1 represents the robot. For simplicity, number 0 is used to represent "nothing". Different locations are labeled as non-negative integers, denoted as loc. And $sort$ denotes the type of the object. The properties ($prop$) of an object include the object's type, location, color and size:

$$\begin{aligned} color &:= \text{white} \mid \text{red} \mid \text{green} \mid \text{yellow} \mid \text{blue} \mid \text{black} \\ size &:= \text{big} \mid \text{small} \\ prop &:= sort(num). \mid color(num). \mid size(num). \mid \text{location}(num, loc). \end{aligned}$$

The robot's state includes its location, the state of the plate and the state of its gripper:

$$robot := \text{location}(1, loc). \mid \text{plate}(num). \mid \text{plate}(0). \mid \text{hold}(num). \mid \text{hold}(0).$$

[1] `http://www.wrighteagle.org/homesimulation/en/competitions.php`

A statement about a scenario description, denoted as *state*, is either a property of an object or a state of the robot: $state := prop \mid robot$.

We assume that human-robot dialogues are spoken in limited segments of natural languages (LSNLs) [5]. A task description specifies a user's task and may consist of three components: goal, constraint, and additional information. They are expressed in a simplified LSNL (actually, a command language). In fact, we set some sub-competitions for a real LSNL, however the results are similar for simplified LSNL. Therefore, we concentrate on the sub-competitions for the command language here.

Formally, a goal is defined as follows.

$$\begin{aligned} goal := {} & \text{give(human, } obj_1\text{)} \mid \text{puton(}obj_1\text{, } obj_2\text{)} \mid \text{goto(}obj_1\text{)} \mid \\ & \text{putdown(}obj_1\text{)} \mid \text{pickup(}obj_1\text{)}, \end{aligned}$$

where

$$\begin{aligned} adj &:= \text{big} \mid \text{small} \mid \text{white} \mid \text{black} \mid \text{red} \mid \text{green} \mid \text{yellow} \mid \text{blue} \\ obj &:= sort \mid adj\ sort \end{aligned}$$

The additional information *info* is defined as follows, which specifies some supplementary information to the initial state of the problem:

$$info := \text{on(}obj_1\text{, } obj_2\text{)} \mid \text{near(}obj_1\text{, } obj_2\text{)} \mid \text{onplate(}obj_1\text{)}$$

A constraint *cons* is defined as:

$$cons := not\ goal \mid not\ info \mid not\ not\ info,$$

which specifies the conditions that must be satisfied during the process of task execution. *not goal* means the action specified in *goal* is forbidden, *not info* means the condition specified in *info* needs to be avoided, and *not not info* means the condition specified in *info* needs to be maintained.

Finally, a task description is defined as a set of *goal*, *cons* and *info*, and a statement $task := goal \mid info \mid cons$.

Scoring Criteria A competition consists of two stages, each containing 40 testing problems. The competing programs are evaluated according to their total scores for all the testing problems in two stages. In the first stage, every task description only contains goals, while constraints and other additional information are used for the testing problems in the second stage.

The score of a competing program gets from a testing problem depends on the number of goals and constraints that the program accomplishes or maintains, as well as the number of primitive actions in the resulting plan generated by the program for the problem. The concrete criteria are as follows:

- Accomplishment of a goal: A goal is considered to be accomplished, if the final state after performing the plan generated by the competing program meets the goal specification.

– Maintaining a constraint: A constraint is considered to be maintained, if it has been satisfied from the initial state to the final one, in other words, every step of the plan's execution meets the requirement of the constraint.

The scoring system is defined as following:

– 10 marks for completing a goal.
– 5 marks for maintaining one constraint.
– −3 marks for each *move* action.
– −1 mark for each primitive action of the rest.

Therefore, the score for a testing problem is computed as: 10× the number of completed goals + 5× the number of maintained constraints − 3× the number of *move* actions − the number of the rest actions.

Like other RoboCup simulation leagues, we also developed a simulator logplayer to play back robot's actions in the visual simulation environment for a test problem, as shown in Figure 1.

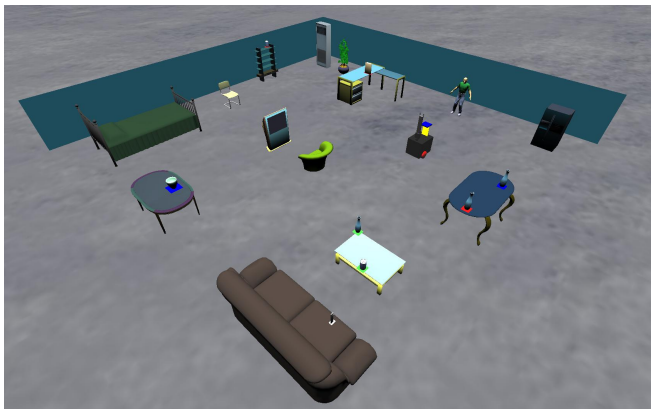


Fig. 1: The Simulator Logplayer for the Simulation Platform

3 Early Competitions

Three competitions based on the testing platform were held in 2009 and 2010[1], respectively. These competitions have similar testing problems. As time goes by, more teams participated and generally they performed better.

5 teams in total participated in the first two competitions on December 2009 and May 2010. All the 80 testing problems are the same in the two competitions. These problems were released after the first competition. All the participants to the second competition knew all the problems from beginning. They also debugged their programs with the problems. In this section, we report the results of the second competition and make comparisons based on the results.

We take three representative competing programs for comparison. They are representatives of the three approaches employed in the competitions and each of them got the highest score in its class. The first one is ours; the competing program is just the high-level part of KeJia robot [4, 5], which is implicated via a nonmonotonic logic programming language called Answer Set Programming (ASP) [2]. This represents the nonmonotonic approach (denoted as NM). The second one is realized with search technology (Search). The third one is based on naive problem-solving approach (PS).

NM approach As presented in [4, 5], the task planning problem in the competition is converted into that of finding an answer set of an ASP program, where the actions of the robot, the scenario descriptions and the task descriptions are represented as ASP rules. Due to the progress on ASP and ASP solvers, as well as the framework problem [11], causality [10], etc, this approach shows many advantages as expected. In particular, there is no difference in handling goals and constraints in this approach, while all the participants adopting other approaches complained about the difficulty of handling constraints. However, efficiency is still the major bottleneck for this approach. In KeJia system, we use iclingo [9] (an incremental ASP solver) to compute answer sets. For a testing problem with 20 portable objects and 14 locations, the system can compute an optimal plan with 12 actions in 5 seconds. More complicated problems may not be solved within the time limit.

Search approach The search approach treats a testing problem as a search problem. The competing program is based on a depth-first search algorithm with some pruning strategies. Firstly, the initial state is acquired from the scenario description and the additional information in the task description. Based on the initial state, the algorithm chooses an primitive action to expand, if the succeeding state meets the requirement of the task, then a plan is computed and it would be stored if it is better than the current best partial plan. If a plan is found, there are not any proper actions to expand, or the search steps are longer than the current best plan, then the algorithm will backtrack to the precious state. Due to the very large state space, strong pruning strategies are needed to ensure that the algorithm terminates in a finite time. But these strategies may exclude the optimal plan—this is the price for the efficiency of the program.

PS approach This approach requires the programmer to predict the detailed solutions for the possible cases of testing problems. A simple strategy is to code a solution for each goal in the task description. The generated plan can be improved by adjusting the order of goals and choosing proper objects. For example, if there are two goals "give" and "puton", the program can choose the objects which are initially at the same location, then an optimal plan can be achieved by holding these two objects at the same time (pick up one and put another on the plate). It is not easy for this approach to handle constraints in the task description. Instead, the constraints were only employed to rule out some forbidden actions. This approach is efficient, but not flexible or reliable. It cannot guarantee to compute the optimal plan for every problem. Re-programming is needed when the domain of the problem changes.

Results There are 14 locations, 8 to 21 different portable objects in the testing scenarios. Initially the robot may have some portable object on its plate or in its gripper. We have run the platform for the second competition on a computer with an AMD Athlon(tm) II X4 620 CPU and a 2GB RAM, the logs and the competing programs can be downloaded from the web site[1].

For a single problem in stage 1, there are 2 to 4 different goals in the task description and optimally it will take 5 to 15 actions to accomplish a task. Note that the difficulty of a testing problem depends on whether or not it contains "related goals". Two goals in a problem are called *related*, if interleaving the execution of actions for them can reduce the total cost (an example is given in Section 5). If goals in a problem are not related, then they can be accomplished separately without loss of efficiency. Based on this observation, we list the results with and without related goals, respectively. The results of stage 1 are shown in Table 1.

Table 1: Results of stage 1 for the 2nd competition

competing program	score for problems without related goals (14)	score for problems with related goals (26)	total score (40 problems)
NM approach	261	460	721
search approach	242	343	585
PS approach	274	410	684

The competing program based on the NM approach returns the optimal plans for 38 problems in stage 1, while the competing programs based on the search and PS approaches find out plans, which may not be optimal, for all problems. For problems without related goals, the NM approach program and the PS approach program compute the same results, except for one problem for which NM runs out of time. For problems with related goals, the NM program can always compute the optimal plans if it completes the task within the time limit, while the PS program returns the plans which are closed to the optimal plans.

For a single problem in stage 2, there are 2 to 4 different goals, at most 5 constraints and 3 pieces of additional information. Optimally, a program will take 5 to 13 actions to accomplish a task. If a problem contains constraints, it requires further reasoning. For example, "pickup(red bottle)" is a goal and "not not on(bottle,table)" is a constraint, which means that "there must be a bottle on the table". Suppose that initially the 'red bottle' is the only bottle on the table. Then the robot should first put another bottle on the table to accomplish the task. Therefore, constraints add another kind of difficulty. The results of stage 2 are shown in Table 2.

The NM approach returns the optimal plans for 39 problems in stage 2, while the search and PS approach find out plans for all problems. The results show that the NM approach works better for problems with constraints if it can complete the planning in time (it runs out of time for one problem with constraints). It is also shown that the competing program by search approach encountered more difficulty in handling constraints.

Table 2: Results of stage 2 for the 2nd competition

competing program	score for problems without constraints (9)	score for problems with constraints (31)	total score (40 problems)
NM approach	133	700	833
search approach	112	552	664
PS approach	120	625	745

For most testing problems in both stages, the NM approach program can find out the optimal plans in 5 seconds, but fails for some complicated problems (need more than 12 actions to accomplish). This indicates that the NM program is "religious" and "cautious". The search approach program returns plans for all problems in both stages, but the pruning strategies rule out the optimal plans for most problems. The PS approach program returns plans for all problems. Although for most problems in stage 1 the results are closed to optimal plans, the gap grows for complicated problems, especially when there are constrains.

Another competition was held on July 2010[1]. The competition uses 80 new testing problems with the similar size of previous problems. There are 11 different teams in the competition. Their results and corresponding approaches are listed in Table 3 and 4. Note that, the results still meet the previous observation.

Table 3: Results of stage 1 for the 3rd competition

competing program	score for problems without related goals (8)	score for problems with related goals (32)	total score (40 problems)
Team A (NM)	156	705	861
Team B (NM)	149	697	846
Team C (PS)	132	654	786
Team D (search)	117	597	714
Team E (PS)	108	542	650
Team F (PS)	131	515	646
Team G (PS)	124	508	632
Team H (PS)	118	464	582
Team I (PS)	20	-77	-57
Team J (PS)	-20	-98	-118
Team K (PS)	-36	114	-150

Table 4: Results of stage 2 for the 3rd competition

competing program	score for problems without constraints (5)	score for problems with constraints (35)	total score (40 problems)
Team A (NM)	87	948	1035
Team B (NM)	69	854	923
Team C (PS)	84	815	899
Team D (search)	74	826	900
Team E (PS)	58	798	856
Team F (PS)	72	739	811
Team G (PS)	76	726	802
Team H (PS)	87	653	740
Team I (PS)	17	266	283

4 The 4th and 5th Competition

In 2011, two competitions based on the simulation platform were held on May[1] and August[2] respectively with more challenging testing problems. Each of the competitions has 12 teams. Different from previous ones, more primitive actions are allotted to the virtual robot and more variables are considered in these competitions. In particular, a new type of objects named "container" is introduced and four new primitive actions become available to the virtual robot.

- $putin(A, C)$: The robot puts object A into container C.
- $takeout(A, C)$: The robot takes out object A from container C.
- $open(C)$: The robot opens container C.
- $close(C)$: The robot closes container C.

Generally, each testing problem involves 30 portable objects and 17 locations, which requires 12 to 23 actions to accomplish the test.

Clearly, testing problems became more challenging. However, most teams still performed well. Despite approaches reported in Section 3, some new approaches are employed in the competitions.

Improved NM approach On top of the NM approach, "macro actions" are introduced as consecutive executions of some original actions. When a plan contains a macro action, it means the robot should execute a sequence of actions at the step. Clearly, using macro actions can reduce the number of actions in a plan, thus improve the efficiency of the original approach. However, the plan contained with macro actions may not be an optimal solution. We can remedy the problem through careful definitions of macro actions.

IDA* Search approach The approach is based on the Iterative Deepening A* (IDA*) search algorithm, which is a variant of the A* search algorithm using iterative deepening to keep the memory usage lower than in A*. The heuristic is essential for the performance of the approach and some pruning techniques are still required for certain cases.

NM plus PS approach The NM approach can compute an optimal plan taking a long time, while the PS approach can compute a plan in shorter time that is not necessarily optimal. This approach combines the benefits of both approaches. It first provides a skeleton of the solution by the PS approach, then fulfills details by the NM approach. However, the solution may not be optimal.

Improved PS approach The approach improves the original PS approach through a much deeper analysis of structures of corresponding testing problems. For each testing problem, the approach creates a directed graph based on the initial and goal locations of related objects. Then, a strategy of solving the problem is chosen based on the structure of the graph.

The results of the 5th competition (similar to the results of the 4th competition) are listed in Table 5. It shows that most teams perform well and the Improved NM approach and the IDA* approach perform better than others.

[2] `http://www.wrighteagle.org/rco/athome/2011/results.php`

Table 5: Results of the 5th competition

Team Name	score for stage 1 (40 problems)	score for stage 2 (40 problems)
Team A(improved NM)	1060	1880
Team B(IDA*)	1061	1850
Team C(NM+PS)	912	1735
Team D(NM+PS)	1003	1691
Team E(improved PS)	954	1672
Team F(improved PS)	844	1671
Team G(PS)	787	1486
Team H(PS)	816	1448
Team I(PS)	588	-
Team J(PS)	435	-
Team K(PS)	357	-
Team L(PS)	0	-

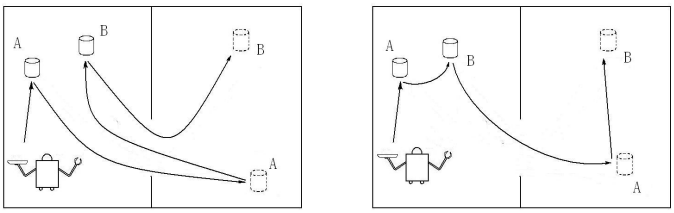
5 Discussion

Since 2010, the RoboCup@Home competition has added a new suit of tests, named General Purpose Service Robot (GPSR) [6, 7]. Different from other tests, GPSR is not incorporated into a story and there is not a predefined set of actions. In the test, the domestic robot is asked to serve user's needs which are specified in English. Note that, the requirements for high-level functions in GPSR are similar to the requirements in simulation competitions reported in this paper.

We believe that, besides underlying robotics techniques, high-level functions are also crucial for future domestic applications of a service robot. In the simulation competitions, the following three issues related to high-level functions are mainly considered.

(1) **Planning for Complicated Tasks** Figure 2 shows an example of a complicated task. Suppose you and your friend are setting in the living room and you ask your robot to fetch two cans of beer from the dining room. This request is a complex task consisting of two related goals, move the first can from the dining room to the living room and move the second from the dining room to the living room. If the robot cannot understand or make use of the relatedness of the goals, it will fetch the cans one by one separately, as shown in Figure 2a. Obviously, it is not necessarily optimal and is typically inefficient. An optimal plan is shown in Figure 2b. This is the way an intelligent robot is expected to do it. In the simulation platform, the optimal plan would get the highest score. Different testing problems in competitions correspond to various complicated tasks in domestic applications.

(2) **From Dialogue Understanding to Planning** An important requirement for a intelligent service robot is to extract knowledge and information from human-robot dialogue, and translate them to task planners, with which the task planners can make use of the knowledge and information to solve new problems and the robots can accumulate knowledge and improve performance. In competitions, we use tasks specified in LSNLs or a simplified LSNL to simulate sentences in the human-robot dialogue.



(a) (b)

Fig. 2: Related goals

(3) **Efficiency Issues** Robots are required to quickly respond to users' utterances. Then efficiency issues become more acute, dialogue understanding and task planning should be terminated in a short time. In competitions, each program needs to return a result in 5 seconds, which is taken as the length of time that users can tolerate.

From the results of the series of competitions, we can see that most teams perform better and better, especially these teams using the Improved NM approach or the IDA* approach. With the accumulation of the five competitions, we can see that the testing problems become more and more challenging. In the 1st competition, a testing problem may contain 14 different locations and 8 to 21 portable objects, and the problem can be solved less than 15 steps. While in the 4th and 5th competitions, a testing problem involves 17 locations and 30 portable objects, and the problem requires 12 to 23 actions to be solved. On the other hand, the performance of participating teams also become better and better. In the first two competitions, only a few teams performed well. While in the last two competitions, most teams can solve almost all testing problems and the differences of their performances are lessening.

6 Conclusion

In this paper, we report five simulation competitions based on a platform for evaluating high-level function of a domestic robot. These competitions focus on the performance of a robot on task planning and dialogue understanding while avoiding the consideration of robots' hardware. From the results of the series of competitions, we can see that more and better approaches have been developed through the competitions, indicating that the competitions are welcome by researchers and students (graduates and undergraduates) and also helpful for promoting research and education on high-level functions of service robots. In addition, we hope this competition will help draw more and more teams to participate in real robot competitions as real robots become available to more and more people.

In the future, we will extend the simulation competition to consider other high-level functions of domestic robots, including coping with dynamic environ-

ments, failure recovery, uncertain information processing, human-robot dialogue during the execution of a current plan, multi-robot scenarios and so on.

Acknowledgments. This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the National Natural Science Foundation of China under grants 60745002 and 61175057. We thank Daniele Nardi, Wei Liu and Fangkai Yang for their help to this work. We are also grateful to the anonymous reviewers for their constructive comments and suggestions.

References

1. Asoh, H., Vlassis, N., Motomura, Y., Asano, F., Hara, I., Hayamizu, S., Ito, K., Kurita, T., Matsui, T., Bunschoten, R., Kröse, B.: Jijo-2: An office robot that communicates and learns. IEEE Intelligent Systems 16(5), 46–55 (2001)
2. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, New York, NY, USA (2003)
3. Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S.: Experiences with an interactive museum tour-guide robot. Artificial Intelligence 114(1-2), 3–55 (1999)
4. Chen, X., Jiang, J., Ji, J., Jin, G., Wang, F.: Integrating nlp with reasoning about actions for autonomous agents communicating with humans. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (IAT-09). pp. 137–140 (2009)
5. Chen, X., Ji, J., Jiang, J., Jin, G., Wang, F., Xie, J.: Developing high-level cognitive functions for service robots. In: Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10). pp. 989–996 (2010)
6. Committee, R.H.T., et al.: Robocup@home: Rules and regulation (2010)
7. Committee, R.H.T., et al.: Robocup@home: Rules and regulation (2011)
8. Ferrein, A., Lakemeyer, G.: Logic-based robot control in highly dynamic domains. Robotics and Autonomous Systems 56(11), 980–991 (2008)
9. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: Engineering an incremental asp solver. In: Proceedings of the Twenty-Fourth International Conference on Logic Programming (ICLP-08). pp. 190–205 (2008)
10. Lin, F.: Embracing causality in specifying the indeterminate effects of actions. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96). pp. 670–677 (1996)
11. Reiter, R.: The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy pp. 359–380 (1991)
12. Tenorth, M., Beetz, M.: KnowRob — knowledge processing for autonomous personal robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-09). pp. 4261–4266 (2009)
13. Wisspeintner, T., van der Zant, T., Iocchi, L., Schiffer, S.: Robocup@home: Scientific competition and benchmarking for domestic service robots. Interaction Studies 10(3), 392–426 (2009)