An Ontology-based Multi-level Robot Architecture for Learning from Experiences

S. Rockel, B. Neumann, J. Zhang
University of Hamburg
{rockel, neumann, zhang}@informatik.uni-hamburg.de

K. S. R. Dubba, A. G. Cohn
University of Leeds
scksrd, a.g.cohn@leeds.ac.uk

Š. Konečný, M. Mansouri, F. Pecora, A. Saffiotti
Örebro University
{sky, mmi, fpa, asaffio}@aass.oru.se

M. Günther, S. Stock, J. Hertzberg
University of Osnabrück
{mguenthe, sestock, jhertzbe}@uos.de

A. M. Tomé, A. Pinho, L. Seabra Lopes
University of Aveiro
{ana, apa, lsl}@ua.pt

S. von Riegen, L. Hotz
HITeC e.V.
{svriegen, hotz}@informatik.uni-hamburg.de

Abstract

One way to improve the robustness and flexibility of robot performance is to let the robot learn from its experiences. In this paper, we describe the architecture and knowledge-representation framework for a service robot being developed in the EU project RACE, and present examples illustrating how learning from experiences will be achieved. As a unique innovative feature, the framework combines memory records of low-level robot activities with ontology-based high-level semantic descriptions.

1 Introduction

In this paper, we give an overview of goals and current work in the project RACE[1] (Robustness by Autonomous Competence Enhancement). The main thrust of RACE is to develop a framework and methods for learning from experiences. The ability to conceptualize stored experiences and to adapt plans and behaviour according to experiences is clearly a desirable asset of intelligent robots. It can help robots to expand their knowledge about a complex world, adapt to changes, and cope with new situations.

To achieve this goal, experiences are recorded as semantic spatio-temporal structures connecting high-level representations, including tasks and behaviours, via their constituents at lower levels down to the sensory and actuator level. In this way, experiences provide a detailed account of how the robot has achieved past goals or how it has failed, and what sensory events have accompanied the activities.

Work in RACE therefore combines research from several communities: (1) an ontology-based multi-level knowledge-representation framework has been devised connecting actuator and sensory experiences with higher-level semantic structures, (2) reasoning facilities for both symbolic and quantitative knowledge are in place to deal with hybrid and diverse knowledge, (3) scene descriptions are enriched by high-level semantic interpretations, (4) learning procedures are being devised exploiting all levels of recorded experiences, and – last, but not least – (5) all this is integrated with a state-of-the-art robot platform (a PR2).

The experimental domain used in RACE is a restaurant environment where the robot serves guests. A typical task is to fetch a mug from a counter, bring it to a guest and place it on the table in the proper position. After several experiences, the robot learns how to deal with guests at tables and positions not encountered before.

In the next section, we present the architecture of the integrated robot system. The robot memory is a central component for storing multi-level factual knowledge in a coherent way, in particular experiences comprising instructions, plans, activities, perceptions, and failures. We then describe the OWL-based ontology structure which is extended by SWRL rules that are connected to a constraint system for temporal, spatial and resource reasoning. We also provide details about the representation of concrete scenarios and experiences.

In Section 3, we present several approaches for learning from experiences which are being developed in ongoing research. As one example, we show how repeated detailed instructions for serving a guest can be generalized to cover new situations.

2 Architecture and Knowledge Representation Framework

In this section, we present the main components of the modular RACE architecture and describe important features of the knowledge representation and reasoning (KR&R) framework. Most components exist as prototypes, the integration is currently underway.

2.1 Architecture of the RACE Robot System

The central piece of the RACE architecture (Figure 1) is the *Blackboard*, the contents of which are similar to what can be found in an ABox in Description Logics. The Blackboard keeps track of the evolution of both the internal state of the robot and the events observed in the environment and is implemented as an RDF database. The other modules for perception, reasoning, planning and execution communicate by

[1]RACE is funded by the EC Seventh Framework Program theme FP7-ICT-2011-7, grant agreement no. 287752.

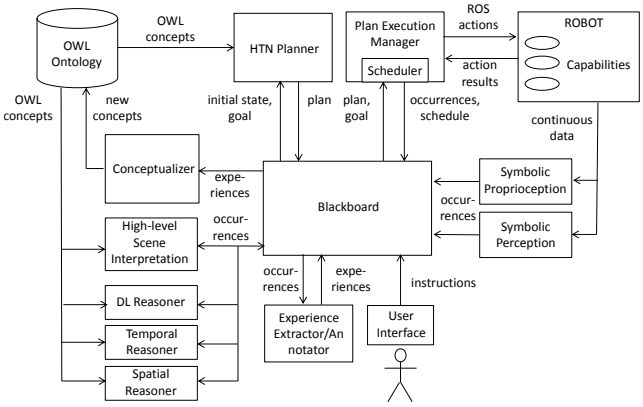

Figure 1: The RACE Architecture.

reading selected types of information from the Blackboard, processing this information and writing back their output. In our architecture, the Blackboard serves two roles: it maintains a consistent representation of occurrences (executed actions, world state propositions, etc.), with begin and end timestamps, from which the current state, as well as past state information, can be derived; and it stores complete experience records (e.g., execution of a coffee serving task), which can be conceptualized later. Details of the experience representation format can be found in Section 2.4.

When a new planning goal is entered by the user, the *HTN Planner* (Nau et al. 2001) queries the Blackboard to build its initial planning state, then writes the generated plan back into the Blackboard. The stored plan includes operator pre- and postconditions as well as the hierarchy of expanded HTN methods. The plan is picked up by the *Execution Manager*, which starts dispatching the planned actions to the robot platform. The *Execution Manager* also performs actions reactively when necessary. During execution, the manager logs the executed actions, as well as success or failure information, to the Blackboard.

The robot used in this project is a PR2, running ROS (Quigley et al. 2009). ROS already provides many capabilities (e.g., for manipulation or navigation) in the form of ROS actions; others will be added as needed. The planning domain specification is aligned to the capabilities of the robot such that each planning operator corresponds to one ROS capability and can be executed directly. Since capabilities run in a tight perception-action loop, some failure cases can be handled locally; if an action fails permanently, the Execution Manager is notified, which will in turn trigger re-planning.

Not all plan failures can be detected locally by the capability itself. For instance, if a bottle is picked up, but slips from the gripper en route to another table, the pick-up action has already terminated successfully. However, the plan execution manager can infer that the bottle has slipped by using proprioception (since the gripper fully closes). By checking the current world state against the operator preconditions, this kind of plan failure can be detected. More generally, the detection of failures requires inference, by which the robot's own expectations are checked for consistency with respect to the observed state of the world. In RACE, this is achieved by invoking ontological and constraint-based reasoning services (described, respectively, in Sections 2.2 and 2.3).

The robot provides continuous data about its own status (such as joint angles) as well as data from its various sensors. Then the *symbolic perception/proprioception* modules discretize this information into symbolic timestamped occurrences; example outputs of these modules indicate whether the robot's gripper is open, closed, or moving, or the qualitative spatial location of a piece of cutlery observed by the robot's RGB-D camera. While the outputs refer to discrete, symbolic entities (such as servingArea1), these entities may have quantitative properties (such as the spatial polygon that servingArea1 represents).

The robot's conceptual knowledge is stored in the *OWL ontology* (Section 2.2). The ontology provides a common representation format from which the domain knowledge of all other reasoners is generated. For instance, the HTN planning domain is extracted from the conceptual knowledge about robot activities. Similarly, the OWL ontology feeds the spatial, temporal and ontological reasoners as well as the high-level scene interpretation. These reasoners enrich the basic experiences in the Blackboard with higher-level semantic information and can also be queried directly by other modules. For instance, if the robot's task is to serve coffees, temporal inference may tell it that the act of delivering coffees must be scheduled before the coffees become cold. Similarly, metric spatial reasoning can endow the robot with the ability to discern exactly where to place a dish when serving (between the fork and the knife of a well-set table).

Background processes responsible for experience extraction and conceptualization support a long-term learning loop. The history of occurrences in the Blackboard is continuously observed by the *Experience Extractor* to detect and extract potentially relevant experiences, based on plan structure, user feedback, pauses, similarity with stored experiences, and conceptualizations and novelty.

Experiences are then used to improve future performance by employing several learning methods. Taking the experiences from the Blackboard as input, the *Conceptualizer* (Section 3.2) modifies the ontology, resulting in more robust and flexible future plans.

2.2 Structure of the Ontology

The robot's conceptual knowledge about the world and the restaurant domain is represented in OWL 2 DL, extended by SWRL rules to express constraints. The ontology is structured into an Upper Model with generally useful concepts (e.g., Occurrence, RobotActivity, PhysicalEntity) and a Domain Model with concepts specific for the restaurant domain. A large part of the domain concepts describes occurrences which may happen as robot or guest activities. As an example, consider the robot activity of moving an object from one place to another, described below in Manchester OWL Syntax:

```
Class: MoveObjectFromTo
EquivalentTo:
RobotActivity
```

```
AND (hasObject EXACTLY 1 PhysicalEntity)
AND (hasFromOn EXACTLY 1 On)
AND (hasToOn EXACTLY 1 On)
AND (hasEmptyHand ATLEAST 1 EmptyHand)
AND (hasGetObjectFrom EXACTLY 1 GetObjectFrom)
AND (hasMoveObjectTo EXACTLY 1 MoveObjectTo)
```

In addition to the properties listed above, the concept inherits the properties hasAgent, hasStartTime, hasEndTime from superclass concepts. Note that n-ary relations with $n > 2$ are reified to be representable in OWL DL: Relation types (e.g., On, At, Before) are defined as concepts with the arguments as properties.

Apart from being embedded in the taxonomical hierarchy of OWL, occurrence concepts can be constituents of compositional hierarchies. For example, MoveObjectFromTo has GetObjectFrom and MoveObjectTo as parts, which are further decomposed until primitive robot activities are reached. Concepts consisting of several occurrences are called aggregates. Compositional hierarchies defined in the ontology are transformed into corresponding hierarchies of planning methods and operators for the HTN planner.

OWL DL allows one to refer to quantitative data types which are used here mainly for temporal and spatial information. But OWL provides only rudimentary ways to express constraints on data types, e.g., on durations or locations. To incorporate constraints into concept definitions, we employ the SWRL rule extension of OWL. The LHS of a rule is used to introduce variable names for concept instances, as shown below for part of the concept MoveObjectFromTo (variables are prefixed by a "?").

```
MoveObjectFromTo(?moveobjectfromto)
∧ hasStartTime(?moveobjectfromto,
              ?moveobjectfromto-starttime)
∧ hasFinishTime(?moveobjectfromto,
              ?moveobjectfromto-endtime)
∧ hasAgent(?moveobjectfromto, ?moveobjectfromto-agent)
∧ hasObject(?moveobjectfromto, ?moveobjectfromto-object)
...
```

The RHS (not shown) is used to express constraints on these variables with constraint names referring to constraints implemented in the constraint system.

2.3 Constraint Processing

Much of the conceptual knowledge of the robot (whether learned or provided by first principles) is relational in nature. It is often qualitative, e.g., knowing that desserts are served after the main course, or that forks and knives should be placed, respectively, on the left and right side of plates. However, this knowledge is useful for the robot if it can be instantiated in observed situations. This entails the need for attaching metric information deriving from observations to qualitative conceptual knowledge. For instance, the robot may need to infer, based on qualitative relations describing well-set tables, where to place a fork in the specific case of a particular table on which a dish is present. This requires identifying an admissible area for placing the fork in the reference frame of the table, given the specific placement of the dish in the scene.

Given these requirements, many of the reasoning services provided by the RACE KR&R framework are based on constraint processing techniques. Specifically, qualitative temporal knowledge is represented as relations in Allen's Interval Algebra (Allen 1984). When applicable, these qualitative representations are augmented with metric constraints (van Beek 1990). This allows the robot to perform useful operations such as abductive reasoning to infer context using temporal relations (Pecora et al. 2012), determining earliest and latest bounds for carrying out a task, and performing temporal execution monitoring.

Reasoning about qualitative temporal relations is achieved through path consistency checking and backtracking search in Allen's qualitative temporal calculus. Metric temporal constraint reasoning occurs through two solvers, namely a Temporal Constraint Satisfaction Problem solver and a Simple Temporal Problem solver (Dechter 2003). Variables in these types of problems represent time points, which allow the capture of information regarding events. Pairs of time points combined with a constraint on their temporal separation allow the representation of intervals, which capture information about actions carried out by the robot and states of the environment (e.g., the minimum and maximum duration of a picking action, or the nominal bounds on the duration of a meal).

Intervals and temporal constraints are also the basis for reasoning about the use of reusable resources. These are resources with a given limited capacity, which is decreased when an activity using this resource is executed. The cumulative use of a resource in any interval of time may not exceed its capacity. The framework implements a scheduling algorithm based on meta-CSP reasoning (Cesta, Oddi, and Smith 2002) which resolves over-consumption of resources by imposing temporal constraints which sequence activities in time. The algorithm, which builds on the framework's simple temporal problem solver, enables the robot to refine the temporal bounds of planned activities so as to achieve resource-feasible plan execution. This is an important capability for a waiter robot — for instance, the maximum weight that can be carried by the robot can be modeled as a limited capacity resource.

Another important aspect of the robot's knowledge is related to spatial relations. The RACE KR&R framework provides qualitative and metric spatial reasoning tools that can be invoked, as for temporal reasoning, to refine the robot's knowledge, refine plan execution with quantitative information, or interpret observations. These include the Region Connection Calculus RCC-8 (Randell, Cui, and Cohn 1992), the Directional Calculus (Goyal 2000), and CORE-9 (Cohn, Renz, and Sridhar 2012) which for rectangular regions (such as object bounding boxes) subsumes and improves the expressiveness of both the above calculi as well as the Rectangle Algebra (RA), a spatial calculus which extends Allen's Interval Algebra to two dimensions (Balbiani, Condotta, and Del Cerro 1999). Augmented spatial constraints in RA which capture spatial relations in metric space are reasoned with the combined use of two metric Allen's Interval constraint reasoners. This allows the robot to leverage, similarly to the temporal case, abductive reasoning for scene interpre-

tation (e.g., answering the question "is the table well set?"), plan refinement (e.g., determining the area in which a dish should be placed given the other objects on the table), and plan execution monitoring (e.g., detecting that it is impossible to place the dish on the table because of clutter.)

2.4 Representing Basic Experiences

Experiences are the main source of information used for learning how to achieve a more robust robot behaviour. They arise from robot activities and can be directly exchanged between the modules through ROS messages as well as stored for future use in the Blackboard using the YAML format.

The YAML experience format consists of a header and a footer, both containing the name of the experience, the name of the corresponding ontology and environment, and a start time and finish time. The body is a list of occurrences structured similarly to ABox entries. Occurrences are not only relevant for experiences, but are the central message format for communication between the Blackboard and other modules, as shown in Figure 1. The Blackboard provides services to insert new or changed occurrences. Thus, experiences can be seen as a collection of occurrences that have been relevant for the performance of robot behaviour.

The following example of an occurrence represents a primitive action of moving the left arm to the side:

```
!ObjectDescriptor
Class_Instance: [MoveArmToSide, moveArmToSide1]
StartTime: [61.142, 61.142]
FinishTime: [66.306, 66.306]
Properties:
   - [hasArm, RobotArm, leftArm]
   - [hasResult, ActivityResult, succeeded]
```

The field `Class_Instance` contains the concept in the ontology and the name of the individual. It is followed by two intervals indicating the earliest and latest start and finish times of the occurrence. By using time intervals, we can also represent constrained or indefinite time points, as pointed out in 2.3. Experiences generated in real-time can have multiple entries for the same occurrence. First, when an occurrence starts, it will be generated with an indefinite (or constrained) finish time, and when it is finished, another entry will be generated, with a definite finish time and possibly additional properties. In the example above, the action of moving the arm has been performed successfully, which is indicated by the set finish time and the `hasResult` property. Note that activity instances such as `moveArmToSide1` can be property fillers of higher-level aggregates.

Different kinds of information can be stored as occurrences inside an experience. First, we have the primitive activities and states of the robot, e.g., motion of the robot base or the position of the robot's arm. Also perception results such as detected objects or guests are stored. Furthermore, experiences are enriched by higher-level descriptions obtained from the HTN and a scene interpretation module. In addition to real occurrences, the HTN plan itself as well as information about success or failure of the execution is included, reflecting the "mental state" of the robot. We will store the complete plan hierarchy from the overall task down to the operators with their pre- and postconditions. When the plan is executed, success or failure information and timestamps for the start and finish time will be added to the single actions, the higher level tasks and the plan itself. Finally, the user will be able to provide instructions to the robot and give feedback on the execution of a task.

3 Experience-Based Learning

We now describe some of the learning approaches which are being explored and will be investigated using the RACE system as experimental platform.

3.1 A Learning Scenario

We first present a simple scenario in the restaurant domain which will be used as a first-year demonstrator in Project RACE, and illustrate representations and learning approaches for dealing with this scenario.

Scenario A: A restaurant floor plan as shown in Figure 2 where the robot knows the position of mug1 on the counter, the position of table1, the position of guest1 north of table1, and the regions for manipulation, sitting and placing. The robot is instructed to move to counter1, grasp mug1, move to the manipulation region west of table1, and place mug1 at the placement region north of table1. The robot is told that this is a ServeGuest activity.

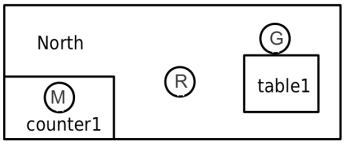

Figure 2: Initial floor plan for counter1, table1, robot Trixi (R), mug1 (M) and guest1 (G) in Scenario A

Scenario B: The same as Scenario A, except guest2 is sitting south of table1 and the robot is instructed to move to the east of table1 and place mug1 at the south of table1. Again, the robot is told that this is a ServeGuest activity.

Scenario C: Guest3 is sitting east of table1 and the robot is simply instructed: Do a ServeGuest to guest3.

The idea is to let the robot discover a generalization of the ServeGuest experiences in Scenarios A and B which will subsume the task in Scenario C.

3.2 Learning about robot activities

This approach can be described as *Constrained Generalization*. The idea is to generalize from experiences and to simultaneously constrain the generalization by commonalities discovered in the experiences. For Scenarios A and B, the unconstrained generalization would describe a ServeGuest for any combination of SittingRegion, PlacingRegion and ManipulationRegion. By discovering common properties or relations in the experiences such as "WithinReach PlacingRegion SittingRegion", the over-generalization can be constrained and will correspond to the desired learning results. For this approach, a large repertory of potentially interesting relations is required, a strategy for choosing common relations is still a research issue. In general, several learning experiences may be required to distinguish, for example,

between irrelevant common guest properties (e.g., colour of garment) and relevant location relations.

The knowledge representation framework of RACE is designed to support Constrained Generalization. Two steps are required: Conceptualization of individual experiences and construction of a constrained least common subsumer (CLCS).

In the conceptualization step, the occurrence instances of an experience are transformed into concepts by abstracting from quantitative time, for example:

```
!ObjectDescriptor
Class_Instance: [On, onA2]
StartTime: [600.020, 600.020]
FinishTime: [780.130, 780.130]
Properties:
  - [hasPhysicalEntity, PhysicalEntity, mugA1]
  - [hasRegion, Region, placingRegionNorthTable1]
⇒
Class: OnA2
SubclassOf: On
AND (hasPhysicalEntity EXACTLY 1 mugA1)
AND (hasRegion EXACTLY 1 placingRegionNorthTable1)
```

Note that concept names begin with upper-case, individuals and roles with lower-case letters. In addition, a new aggregate concept is introduced (in our example ServeGuest) with the top-level activities of the experience as properties (MoveBase, GraspObject, MoveBase, PlaceObject).

The second step (CLCS construction) has two features: (i) Disjunctions are not allowed, hence different individuals are generalized to their closest common superclass:

```
guest1, guest2 ⇒ Guest
placingRegionNorthTable1, placingRegionSouthTable1
                              ⇒ PlacingRegionTable1
```

(ii) Generalizations are constrained by additional properties or relations, to be discovered as commonalities of the underlying experiences.

3.3 Learning about objects and scenes

In order to adapt to different environments and tasks, robots need to categorise physical objects and acquire vocabulary about them. Although the acquisition of visual object categories is a long-standing research problem, the project will approach it from a long-term (or lifelong) perspective and with emphasis on domain open-endedness, i.e. there is no pre-defined set of categories. So, in the above learning scenario, if the robot does not know what a mug looks like, it may ask the user to point to one. Such situation provides an opportunity to collect a training instance for learning. The baseline is a purely memory-based learning approach, in which known instances are stored in memory, and recognition of new instances is based on analyzing similarities with stored instances. An RGB-D camera will be the main source of information for detecting, learning and classifying objects. Objects themselves are represented through a combination of shape models and bag-of-feature models. Finally, acquired object category knowledge is fed into the hybrid knowledge representation and reasoning framework.

In turn, object category knowledge forms the basis for scene recognition. Many scenes have similar structure, or at least elements with structural similarities (the structure of a restaurant, the restaurant table layout, etc.). We will investigate how such knowledge can be acquired in an unsupervised way, through observation and exploration. We will build on our existing work in this area whereby areas of the scene are distinguished semantically via qualitative trajectory analysis and by clustering based on the spatial relationships between qualitatively different parts of the scene. Scene knowledge can also be acquired through human-robot interaction, where the human describes the composition of concrete scenes in terms of objects and spatial relationships. In the case of restaurant table layouts, the robot can realize a certain table layout based on human instructions and then conceptualize the outcome.

3.4 Learning about environment activities

Apart from robot activities, it is also possible to learn different aspects of the world such as the hierarchical spatial layouts of buildings, rooms, tables and chairs and cutlery on a table, object categories based on their functionality and affordance etc. Note that there is also a temporal dimension in some of the concepts that can be learned such as the changes in the cutlery layout as a dinner progresses from starters to desserts. These scenarios are modeled using different spatial relations like RCC-8 etc. among objects as explained in Section 2.3 and the relations are linked using Allen's temporal relations to capture the dynamic changes in the relations. For example the RCC-8 relation between $mugA1$ and $counter1$ in Figure 2 is *NTPP*$(mugA1, counter1)$ (i.e. $mugA1$ is non-tangential proper part of $counter1$). The computed spatio-temporal relational data is stored in the *Blackboard* and will be used for learning.

Learning in this project will be done in supervised and unsupervised settings. In the supervised setting, learning is done by: *learning by observing* and *learning by doing*. In both cases, instructions and feedback from a teacher are necessary and will be provided through the *User Interface* module as shown in Figure 1. In *learning by observing*, the tasks are performed by the teacher, and the robot passively observes the tasks. The teacher gives supervision annotations in the form of concept names: `ServeGuest`, `MakeCoffee` etc. Learning in this case uses spatio-temporal generalization techniques, Inductive Logic Programming and relational graph-based methods (Dubba, Cohn, and Hogg 2010; Sridhar, Cohn, and Hogg 2010). In *learning by doing*, the robot is given detailed instructions (list of goals) to perform a high-level task and then the teacher names what high-level task the robot performed. The instructions also constitute feedback on whether the task was performed correctly or not: they give the robot opportunity to learn and adapt by reasoning why the task went wrong by comparing experiences derived from contents of the Blackboard during the wrongly executed task with that of a correctly executed task.

In unsupervised learning, the robot mines the Blackboard extracting interesting patterns and anomalous sections for future use such as categories of objects, repetitive occurrences and groups of tasks etc.

4 Related Work

Since several diverse research areas are related to this work and due to space limitations, we can only address most relevant previous works in this section.

The idea of autonomous learning from examples has been around from the very beginning of AI (Hunt and Hovland 1963), and symbolic methods using generalization and discovery also date back several decades. However, the context of an embodied robot acting in space and time poses many new challenges. One is to use a formal ontology for knowledge representation for support of reusability and shareability. Our approach is similar to work in RoboEarth, where experiences are also represented using OWL (http://www.roboearth.org/). A major goal in RoboEarth is to obtain a sharable representation of the environment by combining the experiences of many robots, while our focus is on using the experiences of a single robot to improve its performance as a service robot. Another related project is XPERO (http://www.xpero.org/) where the emphasis was on active robot experimentation to enable inductive learning.

There have been several proposals for how to structure ontology-based multi-level KR for robots (Suh et al. 2007; Galindo et al. 2005). In our approach, knowledge is essentially structured around compositional hierarchies where high-level representations build on lower-level constituents, as proposed in research on activity recognition (Neumann and Moeller 2006; Fusier et al. 2007), but we also consider special activity concepts (e.g., resources and goals) as proposed, for example, in (Chen and Nugent 2009).

The connection between symbolic processing and quantitative data has been under discussion in many works. Our approach uses the SWRL extension of OWL to connect to constraints, developed in (Bohlken 2009), and the constraint-based reasoning framework laid out in (Günther et al. 2012).

5 Outlook

In its first year of activities, the RACE project has defined an architecture and KR framework for integrating multiple types of knowledge and multiple reasoners. We believe that these will provide an ideal substrate to address the next grand challenge of RACE: making robots autonomously record, analyse and learn from their experiences.

References

Allen, J. 1984. Towards a general theory of action and time. *Artificial Intelligence* 23(2):123–154.

Balbiani, P.; Condotta, J.-F.; and Del Cerro, L. F. 1999. A new tractable subclass of the rectangle algebra. In *Proc. IJCAI*, volume 1, 442–447.

Bohlken, W.; Neumann, B. 2009. Generation of rules from ontologies for high-level scene interpretation. In Governatori, G.; Hall, J.; and Paschke, A., eds., *Rule Interchange and Applications, Proc. Int. Symposium RuleML 2009*, volume 5858 of *LNCS*, 93–107. Springer.

Cesta, A.; Oddi, A.; and Smith, S. F. 2002. A constraint-based method for project scheduling with time windows. *Journal of Heuristics* 8(1):109–136.

Chen, L., and Nugent, C. 2009. Ontology-based activity recognition in intelligent pervasive environments. *Int. J. Web Inf. Syst.* 5(4):410–430.

Cohn, A. G.; Renz, J.; and Sridhar, M. 2012. Thinking inside the box: A comprehensive spatial representation for video analysis. In *Proc. of KR*.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Dubba, K.; Cohn, A.; and Hogg, D. 2010. Event model learning from complex videos using ILP. In *ECAI*, 93–98.

Fusier, F.; Valentin, V.; Brémond, F.; Thonnat, M.; Borg, M.; Thirde, D.; and Ferryman, J. 2007. Video understanding for complex activity recognition. *Mach. Vis. Appl.* 18(3):167–188.

Galindo, C.; Saffiotti, A.; Coradeschi, S.; Buschka, P.; Fernández-Madrigal, J.; and González, J. 2005. Multi-hierarchical semantic maps for mobile robotics. In *Proc. IROS*, 3492–3497.

Goyal, R. 2000. *Similarity assessment for cardinal directions between extended spatial objects*. Ph.D. Dissertation, The University of Maine. AAI9972143.

Günther, M.; Hertzberg, J.; Mansouri, M.; Pecora, F.; and Saffiotti, A. 2012. Hybrid reasoning in perception: A case study. In *Proc. SYROCO*. Dubrovnik: IFAC.

Hunt, E. B., and Hovland, C. I. 1963. Programming a model of human concept formulation. In Feigenbaum, E., and Feldman, J., eds., *Computers and Thought*. MacGraw-Hill Book Company. 310–325.

Nau, D.; Munoz-Avila, H.; Cao, Y.; Lotem, A.; and Mitchell, S. 2001. Total-order planning with partially ordered subtasks. In *Proc. IJCAI*, volume 17, 425–430.

Neumann, B., and Moeller, R. 2006. On scene interpretation with description logics. In Nagel, H.-H., and Christensen, H., eds., *Cognitive Vision Systems*, volume 3948 of *LNCS*. Springer. 247–275.

Pecora, F.; Cirillo, M.; Dell'Osa, F.; Ullberg, J.; and Saffiotti, A. 2012. A constraint-based approach for proactive, context-aware human support. *Ambient Intell. Smart Environ.* 4(2).

Quigley, M.; Conley, K.; Gerkey, B. P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

Randell, D.; Cui, Z.; and Cohn, A. 1992. A spatial logic based on regions and connection. In *Proc. of KR*.

Sridhar, M.; Cohn, A. G.; and Hogg, D. C. 2010. Unsupervised learning of event classes from video. In *Proc. AAAI*, 1631–1638. AAAI Press.

Suh, I. H.; Lim, G. H.; Hwang, W.; Suh, H.; Choi, J.-H.; and Park, Y.-T. 2007. Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence. In *Proc. IROS*, 429–436. IEEE.

van Beek, P. 1990. *Exact and approximate reasoning about qualitative temporal relations*. Ph.D. Dissertation, University of Waterloo, Waterloo, Ont., Canada, Canada. UMI Order No. GAXNN-61098.