Knowledge-Based Industrial Robotics

Maj STENMARK [1] and Jacek MALEC
Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden

Abstract. When robots are working in dynamic environments, close to humans lacking extensive knowledge of robotics, there is a strong need to simplify the user interaction and make the system execute as autonomously as possible. For industrial robots working side-by-side with humans in manufacturing industry, AI systems are necessary to lower the demand on programming time and expertise. We are convinced that only by building a system with appropriate knowledge and reasoning services, we can simplify the robot programming sufficiently to meet those demands and still get a robust and efficient task execution.

In this paper, we present a system we have realized that aims at fulfilling the above demands. The paper focuses on the ontologies we have created for robotic devices and manufacturing tasks, and presents examples of AI-related services using the semantic descriptions of the skills to help the user instruct the robot adequately.

Keywords. knowledge representation, robot skills, industrial robotics, assembly, service-oriented architecture

1. Introduction

The availability of efficient and cheap computing and storage hardware, together with intensive research on big data and appropriate processing algorithms on one hand, and on semantic web and reasoning algorithms on the other, make the existing results of artificial intelligence studies attractive in many application areas.

The pace of adoption of the knowledge-based paradigm depends on the complexity of the domain, but also on the economic models used and the perspective taken by the leading actors. It may be quite well illustrated by opposing the service robotics area (mostly research-oriented, mostly publicly funded, using open source solutions, acting in non-standardized and not-yet-legally-codified domain) with industrial robotics (application-oriented, privately funded, using normally closed software, enforcing repeatability and reliability of the solutions in legally hard-controlled setting).

When robots are working in dynamic environments, close to humans lacking extensive knowledge of robotics, there is a strong need to simplify the user interaction and make the system execute as autonomously as possible. This also motivates the integration of AI techniques into robotics systems. For industrial robots working side-by-side with humans in manufacturing industry, AI systems are required to lower the programming cost with respect to required time and expertise. We believe that only by building a system with appropriate knowledge and reasoning services, we can simplify the robot

[1] Corresponding author, e-mail: maj.stenmark@cs.lth.se.

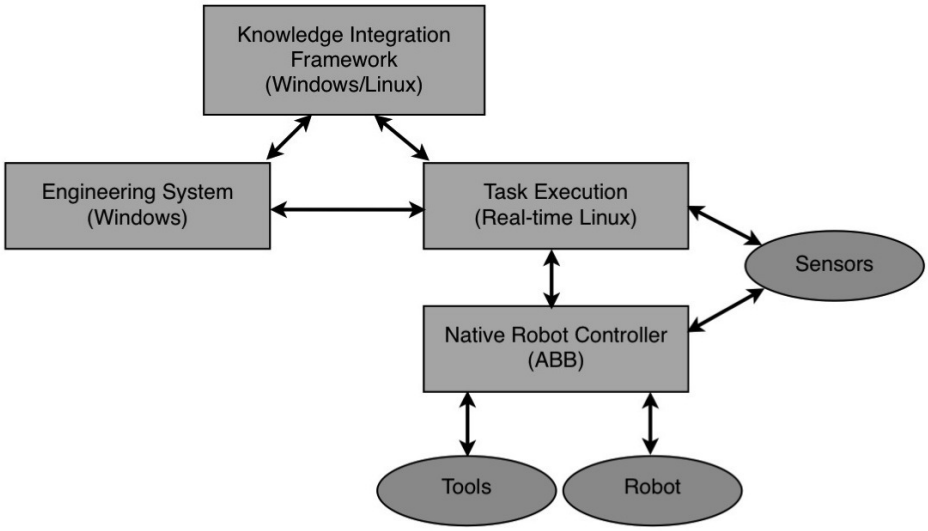


Figure 1. The Knowledge Integration Framework provides services to the Engineering System and the Task Execution. The latter two communicate during deployment and execution of tasks. The Task Execution uses sensor input to control the robot and tools.

programming sufficiently to meet those demands and still get a robust and efficient task execution.

In this paper, we present a system aimed at fulfilling the above demands, focusing on the ontologies we have created for robots and manufacturing tasks and presenting examples of AI-related services that are using the semantic descriptions of the skills to help the user instruct the robot adequately.

The paper is organized as follows: first we briefly introduce the system architecture. Then we describe the knowledge base, and follow with some services provided by the system. Next we introduce the interface towards the user, i.e. the Engineering System, and very briefly describe the program execution environment. Then we describe related research. We conclude by suggesting futher future development.

2. Architecture

The system architecture is very roughly depicted in Fig.1. The Knowledge Integration Framework (KIF) is a server that contains data repositories and ontologies. It provides computing and reasoning services. There are two main types of clients of the KIF server, the Engineering System, which is a robot programming environment, and the robot task execution system.

The task execution system is a layer built on top of the native robot controller. Given the task, the execution system generates the run-time code files, then compiles and executes the code.

The Engineering System uses the ontologies provided by KIF to model the workspace objects and downloads skills and tasks from the skill libraries. Similarly, new objects and skills can be added to the knowledge base by the Engineering System. Skills

that are created using classical programming tools such as various state machine editors (like e.g. JGrafchart [7]), can be parsed, automatically annotated with semantic data and stored in the skill libraries.

The services, described later in the paper, are mainly used by the Engineering System to program, plan and schedule the tasks.

3. Knowledge Integration Framework

The Knowledge Integration Framework, KIF[2], is a module containing a set of robotics ontologies, a set of dynamic data repositories and hosting a number of services provided for the stored knowledge and data. Its main storage structure is a Sesame triple store [9] and a set of services stored in Apache Tomcat servlet container [8][3].

The ontologies we use in our system come from several sources and are used for different purposes. The main, core ontology, `rosetta.owl`, is a continuous development aimed at creating a generic ontology for industrial robotics. Its origins can be traced to the FP6 EU project SIARAS [10], where an ontology of robot skills has been created by the project partners. It has been further modified within the FP6 EU project RoSta (robot standards and reference architectures, `http://www.robot-standards.eu/`), and has been made available as the `rosta.owl` OWL file. Within the FP7 Rosetta project this ontology has been further developed, partially refactored and made available online on the public KIF ontology server `http://kif.cs.lth.se/ontologies/rosetta.owl`. However, this is just the first of a set of ontologies available on KIF and useful for reasoning about robotic tasks.

The ontology hierarchy is depicted in Fig. 2, where arrows denote ontology import operation. We use extensively the QUDT ontologies and vocabularies (Quantities, Units, Dimensions and Types, initiated by NASA and available at `http://www.qudt.org`) in order to express physical units and dimensions. This ontology has been slightly modified to suit the needs of our reasoner.

The core Rosetta ontology (as its predecessors) is focusing mostly on robotic devices and skills. It has been described earlier in [10,11,12]. According to it, every device can offer one or more skills, and every skill is offered by one or more devices. Production processes are divided into tasks (which may be considered specifications), each realized by some skill (implementation). Skills are compositional items: there are primitive skills (non-divisible) and compound ones. Skills may be executed in parallel, if the hardware resources and constraints allow it.

On top of the core ontology we have created a number of "pluggable" ontologies, serving several purposes.

Frames The `frames.owl` ontology deals with feature frames and object frames of physical objects, normally workpieces involved in a task. In particular, the feature frames are related to geometrical locations and therefore the representation of location is of major importance here. The constraints among feature frames are expressed using *kinematic chains*, also introduced by this ontology.

[2]We realize the name coincidence with Knowledge Interchange Format, but as this name has been used for more than five years by now, we have decided to keep it.

[3]Technically speaking, the triple store is also a servlet.

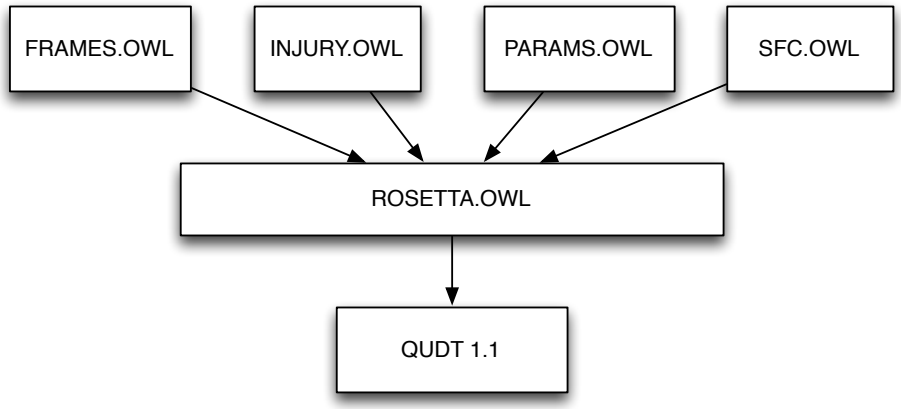


Figure 2. The KIF ontologies used in the Rosetta project.

Injury The `injury.owl` ontology deals with the levels of injury risks when humans and robots cooperate, or at least share common space. The ontology specifies the possible injury kinds, while the associated data, gathered during the Rosetta project [13], are provided as the upper limit values that may be used in computations of injury risks or of evasive trajectories for a robot.

Params Each skill may be parameterized in a number of ways, depending on the granularity level of control, available information or the demands posed on the skill. In order to provide knowledge about skill parameterization for knowledge services (like, e.g. task consistency checking), the `params.owl` ontology describes skills and their mandatory and optional parameters, their units and contraints.

SFC The `sfc.owl` ontology characterizes various behaviour representations using variants of executable state machines (Sequential Function Charts are one of them; the others included are OpenPLC, Statecharts, rFSMs and IML). It also contains the semantic description of several graph-based representations of assembly, like assembly graphs, constraint graphs or task graphs [6], that may also be considered to be behaviour specification, although at a rather high level of abstraction.

This solution illustrates two important principles of *compositionality* and *incrementality*: every non-trivial knowledge base needs to be composable out of simpler elements, possible to be created by a single designer or team without the need to align it with all the other elements. The alignment, or conflict resolution (e.g. inconsistency), should be performed automatically, after plugging the element into the system. So, every "top" ontology should only be forced to adhere to QUDT and ROSETTA ontologies, possibly neglecting other elements existing in parallel.

The incrementality principle ensures that every "top" ontology should be amenable to incremental change without the risk of breaking the whole system. Thus, changes to e.g. Params ontology should not affect the consistency and utility of e.g. SFC ontology. On the other hand, one can imagine situations, where changes in one module (e.g. introduction of a new constraint type between feature frames, described in frames.owl) might facilitate improvements in another (e.g. easier specification of parameters for a given skill, described in params.owl).

Besides storing the ontologies, the triple store of KIF provides also a dynamic semantic storage used by Engineering System to update, modify and reload scene graphs and task definitions. Depending on the kind of repository used, some reasoning support may be provided for the storage functionality. More advanced reasoning, and a generic storage of arbitrary kind of data, is provided by KIF services, described below.

4. Knowledge-Based Services

In order to be able to support several types of AI in the system, KIF provides a set of online services that are accessible over internet. This section describes the services that are implemented in the system. The number of services can easily be extended.

The knowledge base provides storage and reasoning services to its clients. The most basic service it offers are libraries with objects and skills, where the user can upload and download object descriptions and tasks. Some are stored with semantic annotation as triples, e.g. workpieces, scene graphs or skill definitions. Others will be stored as uniform chunks of data without semantically visible structure (e.g. RAPID programs or COLLADA files), although other tools may access and meaningfully manipulate them for various purposes.

Another, more interesting service, is the natural language programming interface [2]. The service takes English sentences as input. The sentences are sent to a natural language parser [3] that outputs a predicate-argument structure for each sentence. The output from the parser is the predicates (verbs) and the arguments (the relevant objects) to the predicates. Each verb has several different *senses* depending on the types and the number of arguments. In the sentence: *Robot, please put the camera socket on the fixture*, the predicate *put* has an actor (robot - arg0) that puts the object (the camera socket - arg1) somewhere (on the fixture - arg2) [4]. If the actor is left out, we assume that it is the robot. The KIF service retrieves the predicate-argument structures from the parser and maps the predicates to existing robot programs and the arguments to station objects. These programs can be edited further and executed on a physical robot or in the virtual environment of the Engineering System.

KIF also provides action planning and scheduling services. The user specifies the task by partially ordering the subgoals in an assembly tree [6]. The planning service can verify the task by checking the pre-and postconditions of each subgoal (skill) and insert actions to fulfill missing conditions.

The scheduling service helps the user assign actions to a system with limited resources. The current implementation of the service is based on list-scheduling. The manipulation skills require different end effectors, e.g., for gripping and screwing. By adding a tool changer to the cell, the robot can change end effectors during the task. The time it takes to change tool is added as penalty on the priority of the actions. When there are multiple arms, one arm can of course change tool while waiting for the other arm to finish during a two-arm manipulation skill. A typical input to the service can be to schedule a partially ordered task on a two-armed robot with three tools and one force sensor. Each action lists its estimated time and the resource requirements, required tool(s) and resources. Given the estimated time to change tools and the number of cycles, the service will output a suggested schedule that minimizes the total time.

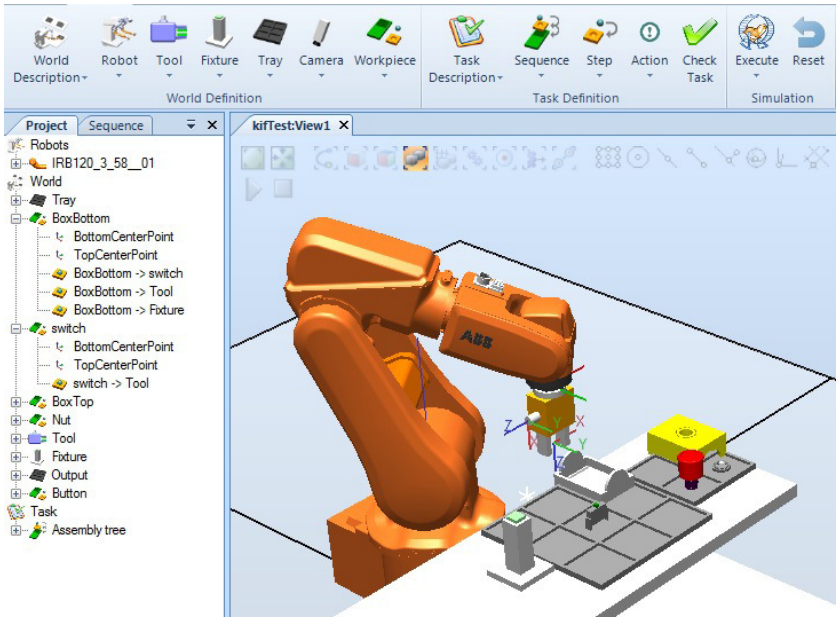


Figure 3. The engineering system is a plug-in the programming environment ABB RobotStudio.

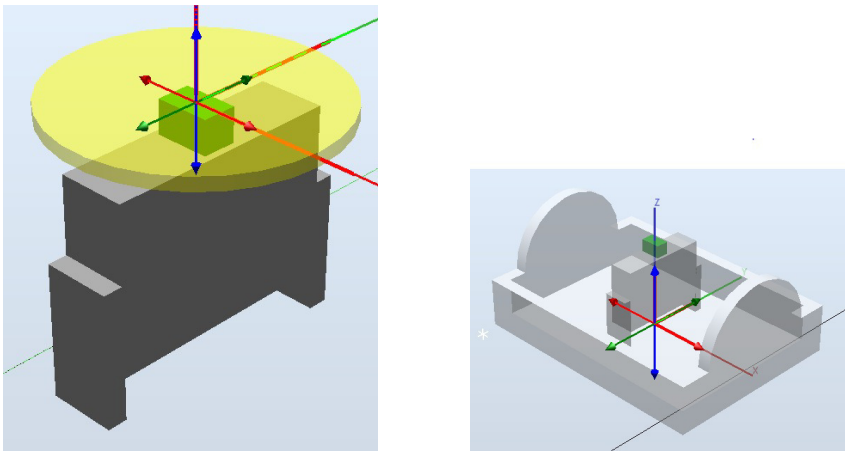
5. Engineering System

The Engineering System is a high level programming interface implemented as a plugin to the programming and simulation IDE ABB RobotStudio [5], see Fig. 3. When creating the station, the objects such as the robot, workpieces, sensors, trays and fixtures, can be generated in the station or downloaded from KIF together with the ontologies.

A physical object is characterized by its *object frame* and a number of *feature frames* related to the object frame, see Fig. 4a. Geometrical constraints are expressed as relations between feature frames, see Fig. 4b.

On the highest level, the task is represented as an assembly graph, which is a partially ordered tree of assembly operations [6]. Fig. 5 displays an assembly graph of a stop button box assembly. In the left subtree, the stop button is placed into the top of the box, then the nut is screwed onto the button. The right subtree represents the assembly of the switch to the bottom of the box. Finally, the root represents the final step where the two subassemblies are assembled. Each assembly operation specifies the desired geometrical relations of the involved objects and the skill type of the operation. The user can download skills from the skill libraries in the knowledge base and adapt the skill parameters to suit the current station.

An example of a simple pick and place sequence is shown in Fig. 6. The program has a nested hierarchy, where the *sequence* step consists of a *pick* and a *place*. These consists in turn of atomic primitives such as motions and actions, e.g., for opening and closing the gripper and locating objects. These types of sequences can be generated from the natural language interface, from the assembly graph, or created manually. When generating the steps, the geometrical relations specified in the assembly graph or between the objects and the tool are used. Each end effector can specify the programs used for its operations



(a) A *feature frame*. (b) A geometrical relation between two objects.

Figure 4. A *feature frame* to the left and a geometrical relation between two objects to the right.

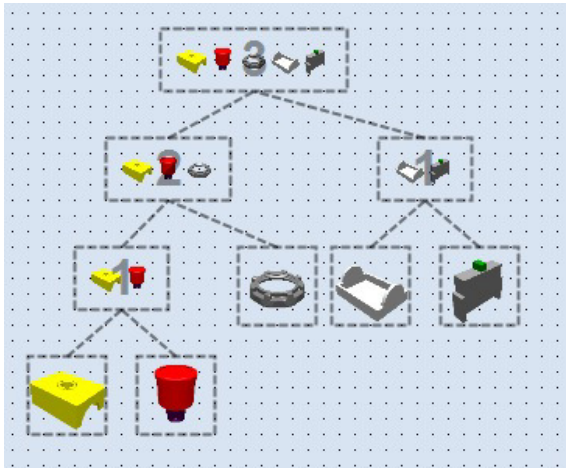


Figure 5. An assembly graph for an assembly of an emergency stop button box.The subtrees are partially ordered, hence there are two parallel assemblies numbered 1.

(open/close or on/off-operations). Thus, the knowledge about the objects is extracted to fill in parameter values of the skills.

When the user has generated a sequence it can be checked for consistency online, using the validity service. A valid sequence can be executed virtually before deploying it on a physical system, and the user can then adapt the sequence until satisfaction.

6. Execution

The sequence displayed in Fig. 6 is the nominal execution. Using the skill descriptions and the constraints between the feature frames in the station, executable state machines

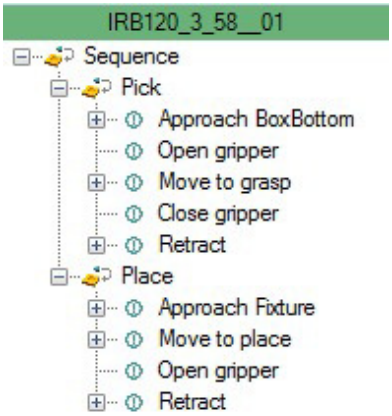


Figure 6. An example of a pick and place sequence.

are generated. Similarly, code for tool functions and simple motions are generated in vendor specific languages. The vendor specific code is executed using the native robot controller, while the more complex sensor-based skills are executed using an external control system [1].

7. Related work

Task representation has been an important area for the domain of robotics, in particular for autonomous robots research. The very first approaches were based on logic as a universal language for representation. A good overview of the early work can be found in [17]. The first autonomous robot, SHAKEY, exploited this approach to the extreme: its planning system STRIPS, its plan execution and monitoring system PLANEX and its learning component (Triangle tables) were all based on first order logic and deduction [14]. This way of thought continued, leading to such efforts as "Naive physics" by Patrick Hayes (see [17]) or "Physics for Robots" [24]. This development stopped because of the insufficient computing power available at that time, but has recently received much attention in the wider context of semantic web. The planning techniques have also advanced much and may be used nowadays for cases of substantial complexity [19], although generic automation problems are usually still beyond this limit.

Later, mixed architectures begun to emerge, with a reasoning layer on the top, reactive layer in the bottom, and some synchronisation mechanism, realized in various disguises, in the middle. This approach to building autonomous robots is prevalent nowadays [15], where researchers try to find an appropriate interface between abstract, declarative description needed for any kind of reasoning and procedural one needed for control. The problem remains open until today, only its complexity (or the complexity of solutions) grows with time and available computing power.

Task description in industrial robotics setting comes also in form of hierarchical representation and control, but the languages used are much more limited (and thus more amenable to effective implementation). There exist a number of standardized approaches, based e.g. on IEC 61131 standards [22] devised for programmable logic controllers, or

proprietary solutions provided by robot manufacturers, however, to a large extent the solutions are incompatible with each other. EU projects like RoSta[4] are attempts to change this situation.

At the theory level all the approaches combining continuous and discrete formalisms may be considered variants or extensions of hybrid systems [20], possibly hierarchical. Hybrid control architectures allow to some extent separation of concerns, where the continuous and real-time phenomena are handled in their part of the system, while the discrete aspects are treated by appropriate discrete tools. Our earlier work attempted at declaratively specifying such hybrid systems, but was limited to knowledge-based configuration [10].

Task descriptions come in different disguises, depending on the context, application domain, level of abstraction considered, tools available, etc. Usually tasks are composed out of skills, understood as capabilities of available devices [16], but the way of finding appropriate composition varies heavily, from manual sequencing in many workflows, via AI-influenced task planning [19], hybrid automata development tools [20], Statecharts [21] and Sequential Function Charts (SFCs) [22], iTaSC specifications [18], to development of monolithic programs in concrete robot programming languages, like e.g. RAPID [5].

There have been several attempts to codify and standardize the vocabulary of robotics. There exists an old ISO standard 8373 requiring however a major revision to suit the demands of contemporary robotics. IEEE Robotics and Automation Society is leading some work towards standardisation of robotic ontologies. In particular, there are first drafts of robotic core ontology [25], although not as developed as the ROSETTA ontology described earlier. Regarding industrial robotics, the work on kitting ontologies originated at NIST [26] may be considered only an early attempt.

8. Conclusions

We have a shown a generic knowledge-based architecture and a couple of examples of its possible use in industrial robotic systems. In particular, we have employed the approach for representing and realizing force-controlled tasks realized by one- and two-handed ABB robots in an industrial setting. The presented generic ontologies are either novel in this work or a derivative of our earlier research. The use of semantic tools and knowledge in industrial robotics is in its early stages with only a few other examples [26]. The ideas have been verified and work well in the ongoing EU-projects PRACE and SMErobotics. We strongly believe that cognition-enabled systems of this kind will gain popularity in the close future.

Future work involves continuing contribution to the IEEE standardization efforts, and aligning and sharing ontologies with other research groups. An on-line documentation of the core ROSETTA ontology is also expected. The number of knowledge-based services should be extended with e.g., online reasoning during execution, geometrical reasoning and integrated path planning and optimization.

[4] www.robot-standards.org

Acknowledgments

The research leading to these results has received partial funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreements No. 230902 (project ROSETTA), No. 285380 (project PRACE) and No. 287787 (project SMErobotics).

The work described in this paper has been done in tight collaboration with many people from the project consortia. The authors are indebted for many fruitful discussions.

References

[1] Anders Blomdell, Isolde Dressler, Klas Nilsson and Anders Robertsson, *Flexible Application Development and High-performance Motion Control Based on External Sensing and Reconfiguration of ABB Industrial Robot Controllers*. In Proc. of ICRA2010, pages 62-66, Anchorage, Alaska, USA, 2010.

[2] Maj Stenmark and Pierre Nugues, *Natural Language Programming of Industrial Robots*, Submitted to International Symposium of Robotics 2013, Seoul, South Korea, 2013.

[3] Anders Björkelund, Bernd Bohnet, Love Hafdell and Pierre Nugues, *A high-performance syntactic and semantic dependency parser*, Proc. of the 23rd International Conference on Computational Linguistics: Demonstrations. Association for Computational Linguistics, 2010.

[4] Martha Palmer, Daniel Gildea and Paul Kingsbury, *The Proposition Bank: An Annotated Corpus of Semantic Roles*, Computational Linguistics, March 2005, Vol. 31, No. 1, pages 71-106. Association for Computational Linguistics, 2005.

[5] ABB RobotStudio, http://new.abb.com/products/robotics/robotstudio.

[6] Jacek Malec, Klas Nilsson, and Herman Bruyninckx. *Describing assembly tasks in a declarative way.* In ICRA 2013 WS on Semantics, Identification and Control of Robot-Human-Environment Interaction, 2013.

[7] JGrafChart, http://www.control.lth.se/grafchart.

[8] Apache Tomcat, http://tomcat.apache.org.

[9] OpenRDF Sesame, http://www.openrdf.org.

[10] Mathias Haage, Jacek Malec, Anders Nilsson, Klas Nilsson and Sławomir Nowaczyk. Declarative-knowledge-based reconfiguration of automation systems using a blackboard architecture Proc. 11th Scandinavian Conference on Artificial Intelligence, Eds: Anders Kofod-Petersen, Fredrik Heintz and Helge Langseth, IOS Press, pp. 163–172, doi: 10.3233/978-1-60750-754-3-163, 2011.

[11] Andres Nilsson, Riccardo Muradore, Klas Nilsson and Paolo Fiorini, *Ontology for Robotics: a Roadmap*, Proceedings of The Int. Conf.Advanced Robotics (ICAR09) , Munich, Germany, 2009.

[12] Anders Björkelund, Jacek Malec, Klas Nilsson, Pierre Nugues and Herman Bruyninckx. *Knowledge for Intelligent Industrial Robots,* Proc. AAAI 2012 Spring Symp. On Designing Intelligent Robots, Stanford Univ., March 2012.

[13] Björn Matthias, Susanne Oberer-Treitz and Hao Ding, *Collision Testing for Human-Robot Collaboration*. In: Safety in Human-Robot Coexistence and Interaction: How Can Standardization and Research benefit from each other?. IEEE Int. Conf. Intelligent Robots and Systems (IROS) 2012.

[14] James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning*. Morgan Kaufmann, 1990.

[15] George A. Bekey. *Autonomous Robots*. MIT Press, 2005.

[16] Anders Björkelund, Lisett Edström, Mathias Haage, Jacek Malec, Klas Nilsson, Pierre Nugues, Sven Gestegård Robertz, Denis Störkle, Anders Blomdell, Rolf Johansson, Magnus Linderoth, Anders Nilsson, Anders Robertsson, Andreas Stolt, and Herman Bruyninckx. On the integration of skilled robot motions for productivity in manufacturing. In *Proc. IEEE International Symposium on Assembly and Manufacturing*, Tampere, Finland, 2011. doi: 10.1109/ISAM.2011.5942366.

[17] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*. Morgan Kaufmann, 1985.

[18] Joris De Schutter, Tinne De Laet, Johan Rutgeerts, Wilm Decré, Ruben Smits, Erwin Aertbeliën, Kasper Claes, and Herman Bruyninckx. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

[19] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning, Theory and Practice*. Morgan-Kaufman, 2004.

[20] Rafal Goebel, Ricardo G. Sanfelice, and Andrew R. Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, 2012.

[21] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[22] IEC. IEC 61131-3: Programmable controllers – part 3: Programming languages. Technical report, International Electrotechnical Commission, 2003.

[23] Markus Klotzbücher and Herman Bruyninckx. Coordinating robotic tasks and systems with rFSM statecharts. *Journal of Software Engineering for Robotics*, 1(3):28–56, 2012.

[24] James G. Schmolze. Physics for robots. In *Proc. AAAI-86*, pages 44–50, 1986.

[25] Joel Luis Carbonera, Sandro Rama Fiorini, Edson Prestes, Vitor A. M. Jorge, Mara Abel, Raj Madhavan, Angela Locoro, Paulo Goncalves, Tamas Haidegger Marcos E. Barreto and Craig Schlenoff, *Defining Position in a Core Ontology for Robotics*, Proc. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 2013.

[26] Stephen Balakirsky, Zeid Kootbally, Craig Schlenoff, Thomas Kramer, and Satyandra Gupta, *An Industrial Robotic Knowledge Representation for Kit Building Applications*, Proc. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 7-12, 2012, Vilamoura, Algarve, Portugal.