Knowledge for Intelligent Industrial Robots

A. Björkelund, J. Malec, K. Nilsson, P. Nugues
Dept. of Computer Science, Lund University
Jacek.Malec@cs.lth.se

H. Bruyninckx
Dept. of Mechanical Engineering, K.U. Leuven
Herman.Bruyninckx@mech.kuleuven.be

Abstract

This paper describes an attempt to provide more intelligence to industrial robotics and automation systems. We develop an architecture to integrate disparate knowledge representations used in different places in robotics and automation. This *knowledge integration framework*, a possibly distributed entity, abstracts the components used in design or production as data sources, and provides a uniform access to them via standard interfaces. Representation is based on the ontology formalizing the process, product and resource triangle, where skills are considered the common element of the three. Production knowledge is being collected now and a preliminary version of KIF undergoes verification.

1 Introduction

Since the beginning of Artificial Intelligence more than fifty years ago, one of the goals (maybe rather dreams) of the discipline was to equip robots with sufficient knowledge and acting capabilities so that they will be able to make decisions and act on their own, but on behalf of their owner, given some goal posed by their human supervisor. This idea starts to materialize nowadays, in particular in *service robots*: companions intended to serve people in different social contexts.

However, the more or less intelligent robots that we envision, usually take the form of a mobile entity, often at least partly humanoid, in order to coexist with humans in their natural environments. Very seldom intelligence is associated with the industrial environments, where the majority of nowadays robots are installed. In a sense, this is natural, as in the industrial setting it is to a much larger extent a matter of cooperation and collaboration between production entities, normally ensured by a centralized control system, while the individual robots remain rather limited in their decision making capacities and autonomy.

The development of knowledge level for autonomous mobile robots is the subject of much attention from the robotics and artificial intelligence communities. Since the times of SHAKEY the researchers have conceived more and more advanced methods to describe the environment of the robots, their goals, their capabilities, and the reasoning necessary to derive an executable action plan leading to some predetermined state of matters. Moore's law allows us to implement nowadays complex representation and reasoning schemes and to achieve efficiency even in case of complex, real-world, albeit limited, domains. Semantic technologies introduced the possibility to exchange knowledge among many individual systems, introducing one more way to find out what to do in a given situation.

This paper presents the idea of applying semantic web technology to increase the level of intelligence of practical, robot-based automation systems, where the necessity to extend the amount of knowledge coexists, or sometimes even contradict, the necessity of guaranteeing accuracy, repeatability, meeting deadlines, error-recovery, etc. In such setting many of the classical knowledge-based solutions need to be reengineered, or at least carefully adapted.

The paper is organized as follows. First we discuss the previous work in the domain, focusing on knowledge representation *in* robots, semantic web techniques, and knowledge representation *for* robots. The next section discusses the specificity of the application domain we focus on in our work. Then we introduce the architecture we are working with. In the following section we describe the necessary set of concepts needed for representing this domain, organized with help of a number of ontologies. Finally, we present the software tools used in our work. The paper ends with conclusions and suggestions for further work.

2 Previous work

2.1 Knowledge Representation for Robots

Knowledge 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 (Brachman & Levesque 1985). 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 (Allen, Hendler, & Tate 1990). This way of thought continued, leading to such efforts as "Naive physics" by Patrick Hayes (see (Brachman & Levesque 1985)) or "Physics for Robots" (Schmolze 1986).

This development stopped because of the insufficient computing power at that time, but has recently received much attention in the wider context of semantic web. The planning techniques have also advanced substantially and may be used nowadays for cases of substantial complexity (Ghallab, Nau, & Traverso 2004), although automation is probably still beyond this limit.

Several alternatives to the symbolic logic-based representation have been developed in the meantime. For a while procedural representations were considered a solution (representing "how?" instead of "what?"), but never have scaled up properly. Rule-based representations simplified the reasoning mechanism while retaining the symbolic representation of facts, and mixed it with purely procedural description of robot actions.

Finally, 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 (Bekey 2005), 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 computing power.

2.2 Semantic Web

The semantic web has recently received much attention from the robotics community. Its roots can be traced back to studies on language semantics and semantic networks (Richens 1956; Quillian 1967). In these pioneering studies, words were pictured as sets of nodes and semantic relationships as labels of the arcs connecting the nodes, such as the *part of* relation in Richens' original paper. For a recent account, see (Sowa 1999). Although the initial intent to use semantic networks for translation has not borne many fruits, the ideas behind it have been revived in the context of the web and the internet. Computational power and storage capacities are available like never before, allowing the modeling, storage, and processing of structures corresponding to real world problems.

Like semantic networks, the semantic web uses a graph to represent concepts, objects, and their relationships. It is organized as a stack of standards with functionalities that goes from character encoding to user applications. Figure 1 shows this stack where the Resource Description Framework (RDF), RDF Schema (RDFS), Web Ontology Language (OWL) are vocabularies to describe the labels of the graph; SPARQL is a query language to extract data from the graphs; and the Rule Interchange Format (RIF) is an ongoing attempt to create a standard way for exchanging reasoning rules.

Using these standardized graph vocabularies (RDF or OWL) and logical tools (SPARQL and RIF), the semantic web enables systems to: encode and interpret data using a rich hierarchical and relational structure, share data with a common format, and extract data and integrate them into applications.

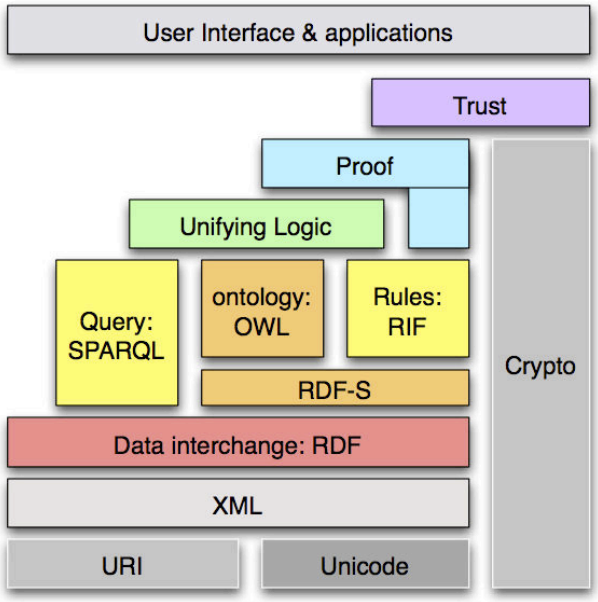

Figure 1: The semantic web stack. After (Bratt 2006).

2.3 Semantic Technology in Robotics

The exploitation of a semantic web approach in robotics may be considered as a paradigm shift: instead of building monolithic, complete systems for robots, why not use a distributed approach with many contributors and many benefactors of the accumulated knowledge. The RoboEarth initiative (Zweigle & Häussermann 2010) is a prominent example of that way of thinking in the area of service robotics.

The tasks expected to be done by a robot define the complexity of the representation. Preprogrammed tasks require no thinking, just acting and reacting to changes. More autonomy and more sensory power requires more reasoning to be done on-line, while acting or just prior to it, as opposed to at the design and programming time. The paradigm shift occurs right now.

Numerous ontologies are created to support such knowledge-intensive systems. Ontologies are just one (but crucial) part of the picture. There have been several attempts to codify production knowledge in form of suitable ontologies (and associated tools). (Lastra & Delamer 2008) provides a relatively recent overview of this expanding field. One of the interesting early attempts, however focusing on collaboration issues in the design process, has been described by (Kim, Manley, & Yang 2006). We have earlier developed an ontology for manufacturing, with focus on devices embodying skills (Persson *et al.* 2010; Haage *et al.* 2011). However, there is no consensus on how this development should proceed.

3 Architecture

Knowledge representation by itself is an activity that might lead to rather theoretical results on representability, com-

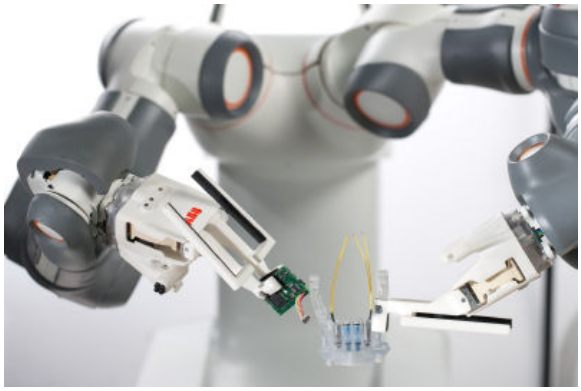

Figure 2: The ABB FRIDA robot.

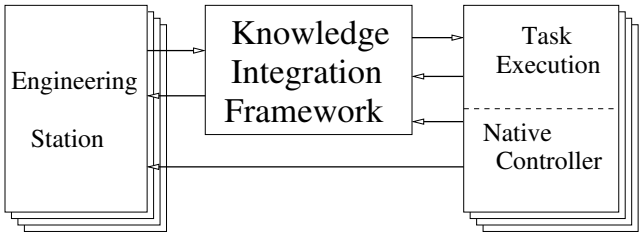

Figure 3: Architecture of the proposed approach.

plexity or expressivity, might entail analysis of reasoning algorithms, or might be considered as a service for other activities leading to constructing a knowledge-based system of some kind. This paper is guided by the latter; we adapt our investigations to the needs and constraints imposed by a concrete system, a co-worker robot based on FRIDA platform, see Fig. 2, that we are building and will demonstrate in 2012.

As mentioned above, we are focusing on manufacturing domain, in particular on robotized, sensor-equipped work-cells where there is a need to efficiently reason about changing the behavior of the system depending on time-varying production demands. One can imagine production of season cookies, assembly of car lamps, or assembly of industrial electronics as cases when varying orders enforces fast reconfiguration of the production equipment. A similar reconfiguration problem arises when some particular production line is to be copied (with some local modifications, e.g. using other types of sensors or robots) to another site.

The architecture of the system is divided into three areas (see Figure 3). In the center there is Knowledge Integration Framework[1] (KIF): a knowledge base serving other subsystems. KIF is not envisioned as a necessarily monolithic unit, but acts as if it were one. Its role is to provide knowledge on any relevant topic to whoever requests it. Primarily we see two kinds of clients that might wish to interact with KIF: engineering stations and cell controllers.

An engineering station (ES) is the main user interface for human users of a system. It is used to define a production cell, its configuration, geometry, resources, connectivity, etc. It is also normally used to develop the work-cell control program, including simulations and possibly virtual execution of the code. Normally there are at least two kinds of users of an ES: those who describe the cell (once in a while, typically system integrators) and those who use it daily (often accessed, but limited to task redefinition and cell reconfiguration, typically technicians). These types of users possess different kinds of knowledge that eventually needs to be stored in KIF if we want it to be reused.

The native controller, together with the actual physical installation, form another kind of "user" for the knowledge available in KIF: they use it for concretizing high-level descriptions provided by the human operators into actual control programs run on real hardware available in the cell. The configuration process might be automatized to some extent, but usually even here a human support is normally needed in order to resolve all ambiguities.

To allow a higher level task execution control, we have augmented our system with a *Task Execution* (TE) software layer. This additional layer has an online connection to the native robot controller and is responsible for realizing and supervising the task on the physical system, interacting with potential external sensors that are not easily integrated with the native system, like camera or force sensor, and able to relay valuable "experience" (such as adapted parameter values, or encountered errors) upstream to KIF.

The interface towards the TE layer is generic, intended to serve stations built with different kinds of hardware and software. On the other hand, interaction with the native robot controller and additional devices and sensors is inherently hardware-specific, and the TE layer needs to implement the functionality to interface with them accordingly. Interaction here includes possibility of supervision and parameter settings, gathering behavioral information such as log data produced by the devices, support for experimentation and semi-automatic adaptation of parameter values, etc. Some of these functionalities use dynamically created code implementing particular communication channel.

Normally there will be many ESs connected to the KIF. In principle, we may expect one ES per one controller/work-cell, but other configurations are possible as well. KIF is supposed to serve as a knowledge storage and exchange database, both for knowledge accumulation (including learning, but also assimilating tacit knowledge) as well as knowledge reuse in various contexts.

It is also important to observe that KIF is not expected to be an omnipotent central system supervising every activity and providing solutions to all problems. The ultimate goal of this architecture is to allow the reusable part of production knowledge to become stored, organized and easily accessible by many users and for many work-cells. We expect that experience will let us draw the borderline between the generic knowledge useful in many situations, and specific setup details relevant only in some particular production case.

As KIF is a repository of knowledge gathering its contents from various sources, there must exist appropriate transfor-

[1] We realize the possible confusion with *Knowledge Interchange Format* (Genesereth 1990), but prefer to keep the name anyway.

mations between the format used by the sources and the representation used by KIF. We have adopted RDF as the storage format (see section 6) and introduced custom-built translators for the data coming from engineering stations. In particular, we employ the Automation ML (Drath 2010) (AML) standard as one of the languages for transferring information about robotic work-cells. AML is suitable for both encoding the static information about the cell (devices, their arrangement, their connections) as well as discrete behaviors encoded as transition systems. Device-specific code is stored in native formats, with only references to the higher-level representations.

4 Domain

There have been several attempts to provide representation for all kinds of physical phenomena that intelligent systems might need to deal with. However, the lesson of CYC (Lenat, Prakash, & Shepherd 1985; Guha & Lenat 1990) shows that such an endeavor is enormous, requires decades and does not necessarily produce satisfactory results.

Therefore, we limit our work to robot-based automation as a sufficiently complex domain for being challenging, but also limited enough to allow non-trivial results to be achieved. In order to obtain meaningful results a substantial amount of consistent knowledge has to be accumulated in the system. Some effort has already been made in this direction in our previous work (Malec *et al.* 2007; Nilsson *et al.* 2009), where manufacturing domain has been first addressed, while the current research focusses mainly on assembly processes. This way we expect to build a substantial knowledge base useful in many areas of the robot-based production.

Sensor-based manufacturing is a particular challenge, because the world is so complex that faithful representation of its state is impossible; therefore some simplification needs always to be made. Representing the sensing process even in case of limited application domain is a very hard problem. In the context of this paper it is considered as a side issue, while most of our work is devoted to reaching satisfactory conclusions on behalf of the users of automation systems.

We have employed the usual approaches to simplifying the problem, namely discretization in space, time, and state, whenever possible. The behavior of sensing devices is normally captured by FSMs, abstracting away the continuous models. However, we make sure that the representation is extensible and modular so that in case when a simplification does not work in some context, an appropriate continuous symbolic model may be inserted and used as needed (Björkelund *et al.* 2011b).

5 Ontology

The conceptual structure employed in our approach is based on the so-called production (PPR) triangle: *product*, *process*, *resources* (Cutting-Decelle *et al.* 2007) (see Figure 4). The *workpieces* being manufactured are maintained in the product-centered view. The manufacturing itself (i.e., the process) is described using concepts corresponding to different levels of abstraction, namely *tasks*, *steps*, and *actions*.

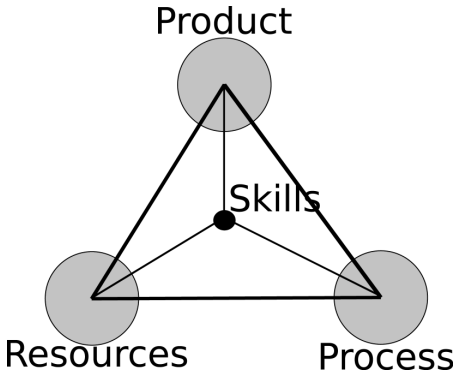

Figure 4: The PPR model, with skills as common coordinating points for the three views.

Finally, the resources are materialized in *devices* (capable of sensing or manufacturing). The central notion of *skill* links all three views and is one of the founding elements of our representation.

In case of robot-based production system, skills may be defined as *coordination of parametrized motions*. This coordination may happen on several levels, both sequencing (expressed, e.g., via a finite state machine or a similar formalism), configuring (via appropriate parametrization of motion) and adapting (by sensor estimation). On top of this approach, based in our case on *feature frame* concept (De Schutter *et al.* 2007), we builds a set of reasoning methods related to task-level description, like e.g., task planning.

The product view in our approach amounts to defining what needs to be done with the workpiece or workpieces. In particular, it is of utter importance to describe the goal of the whole manufacturing process, otherwise no algorithm will be able to meaningfully reason about this process. This goal may be defined explicitly, using some suitable logical formula, but may also be given as a CAD drawing depicting the final state, or may be spelled out using constraints involved in production. Yet another possibility is to describe the transformations of the product that need to be done on the way.

As an example, the *assembly graph* structure has been chosen as a suitable representation for the assembly precedence constraints. In itself, it does not lead to an executable robot program. In order to realize that goal, the following "platform-specific" information must be added:

- gripper(s), and other tooling (screwdrivers, etc.)
- fixture(s)
- robot(s) properties: accuracy, control mode(s)
- sensor(s): force, vision, distance sensor, etc.

Each specific combination of these can lead to a different *assembly scenario*; each of those scenarios, in turn, must deal with a range of uncertainty in the mentioned properties.

The process view describes *how* to achieve the product transformations named in the product view above. To describe the transformations one has to have an appropriate

vocabulary. In our approach, limited to robotized manufacturing, we refer to the following concepts:

motion Continuous time and continuous space activities of robot(s), each moving the robot's tool(s) in a "specified way" until a "termination condition" is reached that indicates that a specified contact situation between two objects in a (sub-)assembly is reached.

action Does not involve robot motions, but other activity like communication, signal processing or decision making, to get more information about the world.

task The fully platform-independent specification of the manufacturing scenario. It contains references to assembly and contact graphs on one side, and to the workpiece descriptions on the other. However, there is no specific, platform-related information available in the task description yet.

Tasks form a hierarchical structure, with sub-tasks decomposable into lower-level task structures.

skill A *finite state machine* (FSM), in which each state runs one single of the above-mentioned motions, and the termination condition of the motion in that state gives rise to a state transition. The FSMs always start in the init state. One FSM can have multiple final states, of which *FullyAssembled* is one; other final states are the envisioned error conditions. The discrete states in a skill are a superset of the states in the task's assembly graph, or rather, the task is also an FSM but with many different hierarchically nested skill FSMs.

Finding out how to perform the transformations is inherently a search problem, usually spelled out as automatic planning, scheduling, or a configuration problem. Depending on how the steps are described and abstracted (operators, activities, or services) different representation languages may be used but the reasoning mechanism in the bottom is the same – an efficient planning algorithm.

The *resource* view is the knowledge that deals with how to allocate and schedule the physical, computational and communication resources needed to execute the task. This knowledge is usually spelled out as skills, linking together devices with processes that are realized (implemented) using them. Skills encode knowledge about possible realisations on (possibly) different hardware and about useful parametrisations necessary to achieve a given objective.

We continue to develop an ontology (understood here technically as a knowledge structure expressed in OWL language) that includes all the notions defined above, in particular all elements of the PPR triangle. But this is just the first step towards providing the encompassing reasoning system with some knowledge about what is reasonable and what is not, in context of automation domain. Ontology only offers a static structure; taxonomy and simple contraints, while the reasoners need much more. This knowledge has to be conveyed using some appropriate representation. We have considered so far:

1. rules (in the RIF variant) and an adequate reasoner to exploit them.
2. behaviors encoded in external formalisms (understood not by KIF itself, but by other tools, like simulators, visualizers, modeling tools, etc.)
3. declarative knowledge suited for specialized reasoners (e.g. mathML and some dedicated solver coupled to KIF capable of using this representation for manipulating models expressed as mathematical formulae).

Given the representations discussed above we may define reasoning as transformation of knowledge from one representation to another. In our case, we have a number of abstraction levels to deal with (continuous, discrete, symbolic) and a number of detail levels within each kind of representation (e.g. similar transition systems, but with differing semantics, are used for describing assembly constraints, tasks, and skills).

6 Tools

There are two kinds of software tools related to KIF: those used for implementing its functionality and those used to interact with the knowledge base and to exploit its contents for engineering purposes.

KIF is implemented using an RDF triple store. We have chosen to use the Sesame RDF repository (openRDF.org 2010). Sesame (with extensions) allows for RDFS and OWL inferencing repositories stored either in memory, on file, or in a database system. Additional generic and specialized reasoners can easily be employed and served by other servlets accessible via the same servlet container, to allow for easy access from the same HTTP node. In particular, we use Jena2 to perform rule-based reasoning in KIF.

Another set of tools is related to the usage of KIF. On the engineering station side we have so far used ABB RobotStudio software as an access and visualisation tool, however the access API is sufficiently generic to allow other, possibly open source, software to be used instead. On the controller side we use so far custom solutions adapted to the hardware (in particular ABB IRb and FRIDA robots, and KUKA manipulators) used for testing our ideas.

We are evaluating our design and filling it with assembly knowledge suitable to support engineers in the work-cell setup process (Björkelund *et al.* 2011a).

7 Conclusions

In this paper we have presented an approach to creating a knowledge repository for the manufacturing domain. Its contents are created by systems engineers describing automation devices and production skills, both generically and in the context of concrete work-cells and stations, as well as is filled from experience. The repository also possesses knowledge about workpieces and appropriate methods for processing them.

The knowledge is exploited for designing and parametrising work-cells intended to be used for similar purpose as the already realized ones, but possibly built using different hardware (sensors, tools, and robots) or software. The task instantiation algorithm is using existing knowledge, but also may provide feedback from the human user in case when another solution is decided to be advantageous - this way the

knowledge base gets extended not only by typical machine learning from data, but also by being told by experts.

The proposed architecture includes a software component that acts as a bridge between the knowledge repository and the native system where the task should be carried out. It enables a higher level task execution control that generically interfaces with the knowledge repository, while at the same time interconnects the different hardware of the production cell, where the native robot controller is just one amongst many devices used in the realisation of the task.

The results obtained so far are preliminary and much work is necessary to reach the expected level of knowledge base contents, useful in practice. The reasoning algorithms are rather simple as well as the ontology we have developed. We are performing now first practical experiments on portability among different, geographically distributed hardware setups, to be used on different brands of hardware without necessity of intervention by humans.

Acknowledgments

The research leading to these results has received partial funding from the European Union's seventh framework program (FP7/2007-2013) under grant agreement No. 230902.

References

Allen, J.; Hendler, J.; and Tate, A., eds. 1990. *Readings in Planning*. Morgan Kaufmann.

Bekey, G. A. 2005. *Autonomous Robots*. MIT Press.

Björkelund, A.; Edström, L.; Haage, M.; Malec, J.; Nilsson, K.; Nugues, P.; Robertz, S. G.; Störkle, D.; Blomdell, A.; Johansson, R.; Linderoth, M.; Nilsson, A.; Robertsson, A.; Stolt, A.; and Bruyninckx, H. 2011a. On the integration of skilled robot motions for productivity in manufacturing. In *Proc. IEEE International Symposium on Assembly and Manufacturing*. doi: 10.1109/ISAM.2011.5942366.

Björkelund, A.; Malec, J.; Nilsson, K.; and Nugues, P. 2011b. Knowledge and skill representations for robotized production. In *Proc. 18th IFAC World Congress*.

Brachman, R. J., and Levesque, H. J., eds. 1985. *Readings in Knowledge Representation*. Morgan Kaufmann.

Bratt, S. 2006. Semantic web and other W3C technologies to watch. `http://www.w3.org/2006/Talks/1023-sb-W3CTechSemWeb/W3CTechSemWeb.pdf`. Site accessed on November 26, 2009.

Cutting-Decelle, A.; Young, R.; Michel, J.; Grangeland, R.; Cardinal, J. L.; and Bourey, J. 2007. ISO 15531 MANDATE: A product-process-resource based approach for managing modularity in production management. *Concurrent Engineering* 15.

De Schutter, J.; De Laet, T.; Rutgeerts, J.; Decré, W.; Smits, R.; Aertbeliën, E.; Claes, K.; and Bruyninckx, H. 2007. 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.

Drath, R., ed. 2010. *Datenaustausch in der Anlagenplanung mit AutomationML. Integration von CAEX, PLCopen, XML und COLLADA*. Springer.

Genesereth, M. R. 1990. Knowledge Interchange Format Version 2.0, Reference Manual. Report Logic-90-4, Computer Science Department, Stanford University. Available at http://logic.stanford.edu/kif/kif.html.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning, Theory and Practice*. Morgan-Kaufman.

Guha, R. V., and Lenat, D. B. 1990. CYC: A midterm report. *AI Magazine* 11(3):32–59.

Haage, M.; Malec, J.; Nilsson, A.; Nilsson, K.; and Nowaczyk, S. 2011. Declarative-knowledge-based reconfiguration of automation systems using a blackboard architecture. In Kofod-Petersen, A.; Heintz, F.; and Langseth, H., eds., *Proc. 11th Scandinavian Conference on Artificial Intelligence*, 163–172. IOS Press. doi: 10.3233/978-1-60750-754-3-163.

Kim, K.-Y.; Manley, D. G.; and Yang, H. 2006. Ontology-based assembly design and information sharing for collaborative product development. *Computer-Aided Design* 38:1233–1250.

Lastra, J. L. M., and Delamer, I. M. 2008. Automation 2.0: Current trends in factory automation. In *6th IEEE International Conference on Industrial Informatics*. IEEE.

Lenat, D.; Prakash, M.; and Shepherd, M. 1985. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottleneck. *AI Magazine* 6(4):65–85.

Malec, J.; Nilsson, A.; Nilsson, K.; and Nowaczyk, S. 2007. Knowledge-based reconfiguration of automation systems. In *Proc. of the IEEE Conference on Automation Science and Engineering*. IEEE. 170–175.

Nilsson, A.; Muradore, R.; Nilsson, K.; and Fiorini, P. 2009. Ontology for robotics: a roadmap. In *Proc. 14th International Conference on Advanced Robotics*.

openRDF.org. 2010. Sesame. `http://www.openrdf.org`.

Persson, J.; Gallois, A.; Björkelund, A.; Hafdell, L.; Haage, M.; Malec, J.; Nilsson, K.; and Nugues, P. 2010. A knowledge integration framework for robotics. In *Proc. International Symposium on Robotics ISR 2010*.

Quillian, M. R. 1967. Word concepts: a theory and simulation of some basic semantic capabilities. *Behavioral Science* 12(5):410–430.

Richens, R. H. 1956. Preprogramming for mechanical translation. *Mechanical Translation* 3(1):20–25.

Schmolze, J. G. 1986. Physics for robots. In *Proc. AAAI-86*, 44–50.

Sowa, J. F. 1999. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove: Brooks Cole Publishing Co.

Zweigle, O., and Häussermann, K. 2010. General architecture of the RoboEarth. In *Proc. IROS 2010 RoboEarth Workshop*.