Advances in Cognitive Systems 1 (2012) 73–81 Submitted 1/2012; published 7/2012

When Do Robots Have to Think?

Claude Sammut CLAUDE@CSE.UNSW.EDU.AU
School of Computing Science and Engineering, University of New South Wales, Sydney, Australia

Abstract

An intelligent robot would seem to be an obvious place to implement a cognitive architecture, yet most robotics applications generally have complex software systems that bear little resemblance to cognitive systems. Furthermore, architectures are usually heavily engineered for specific applications. In this paper, we discuss why general architectures that include deliberation and learning are rarely used in robotics and we pose some challenges for the designers of cognitive architectures to make them more suitable for robotic applications.

1. Introduction

Brady (1985) describes the field of intelligent robotics as being "concerned with the intelligent connection of perception to action". It would seem obvious, given the complex nature of a robotic system, that a cognitive architecture should govern the design of the robot's software. If the robot is expected to achieve human-level intelligence, one would also expect that the architecture to include symbolic reasoning. However, with a few exceptions (Albus, 1997; Albus et al., 1989; Bonasso et al., 1997; Ferrein & Lakemeyer, 2008; Hanford et al., 2009), this is rarely the case.

Why is this so? Is it that high-level reasoning has not been necessary in robotic applications so far? Is it that cognitive architectures are lacking in some important capabilities? Is it due to cultural differences between robotics researchers and those interested in cognitive architectures? The answer to all of these questions is "yes" and we will discuss why in the following sections.

Progress in robotics research is somewhat analogous to evolution, beginning with simple mechanisms that have primitive perceptual and motor abilities, almost no representation of the world and extremely simple behaviours. As in the evolutionary development of animals, it necessary for a robot to develop perceptual and motor skills to be capable of doing anything at all. So it should not be surprising that a great deal of research has been devoted to these areas. With improving perceptual and motor skills, robots have been deployed in increasingly complex tasks. However, both hardware and software are usually highly specialised for the task. In part this is due to the hardware requirements of the application but it also because the behaviours of the robot require extensive engineering for the application. Present day robots are capable of some flexible behaviour, but if circumstances stray too far from those anticipated by the human designers, they are usually incapable of adjusting. In addition to not being able to reason about their own behaviour, they are mostly incapable of describing or explaining it, thus limiting their ability to communicate effectively with humans. Like human evolution, perhaps the next step in robot evolution is to achieve the kind of intelligence that enables a robot to learn to perform new tasks and adapt to changing environments.

An example of state-of-the-art robot technology is Google's self-driving car (Thrun & Urmson, 2011). This is capable of the remarkable feat of driving safely in urban traffic, but to do so requires careful preparation. For example, prior to autonomous driving, the car must be driven around a location manually so that the system can build very accurate maps that let it navigate and distinguish moving objects from fixed objects. Carefully designed heuristics are programmed to control behaviour at stop signs, turning across traffic, avoiding pedestrians, etc. This kind of hand-crafting of behaviours may be economical in applications such as driving, because there are many millions of potential users. It is much less economical in low volume applications, so one would still like a robot to be able to reason about and adapt to novel situations by itself.

It should be noted that the self-driving car has only become feasible in the last few years because of enormous leaps in sensor technology. For example, the Velodyne (2012) laser imaging system can create a 360° point cloud in real time, which greatly simplifies navigation and obstacle avoidance. In the DARPA Urban Challenge (DARPA, 2007), which was a precursor to Google's car, teams that did not have a sensor capable of a 360° field of view were required to do much more reasoning about objects and what happens to them when they move out of the field of view. This is quite difficult and is an example where improved perception can obviate the need for reasoning. One explanation for why robotics researchers are reluctant to put much effort into high-level reasoning is that they expect improvements in hardware technology will remove the need to reason about many things, as is the case with 3D laser imaging. This attitude should not be surprising. The large human cortex that gives us the ability to plan, use language, and th alike, is a late addition in evolution. Most of the brain's development has been devoted to sensory-motor skills. Thus, one of the issues we discuss is how high-level reasoning can take advantage of the machinery developed for low-level skills.

In the following sections, we consider the advantages and disadvantages of high-level representation and reasoning for robots. We examine the limitations of relying too heavily on low-level skills and we will discuss where the balance should be struck between skill and reasoning systems. We conclude by looking at what aspects of cognitive architectures need further development to be more useful in robotics.

2. Intelligence Without Representation

Planning systems for robots are often characterised as being *reactive*, *deliberative* or some combination of the two. By reactive, we mean a system that uses something similar to a classic feedback loop to achieve a goal such as driving a specified distance or moving until coming into contact with an object. That is, the planner has a simple but fast mapping from sensor measurements to motor commands that is executed repeatedly to achieve a specified goal. A reactive system generally does not try to make predictions about the future. It just responds to the current situation, whereas a deliberative planner makes projections into the future, creating a sequence, or partial sequence, of actions needed to achieve a goal. Deliberative planning can be more flexible than reactive control but it is generally slower.

Perhaps the earliest example of a robot architecture that combined reactive and deliberative control was Shakey (Fikes et al., 1972). Although this work is best known for introducing the STRIPS planner, the software architecture also included a reactive component. Once the planner

had generated a sequence of actions, those actions then had to be executed by the hardware. This was controlled by a reactive system. Shakey was truly impressive, given the hardware available at the time, but even taking into account the speed of computers in the 1970s, the architecture could not respond rapidly if the environment were dynamic. In part, this was due to its sense-plan-act cycle, shown in Figure 1.

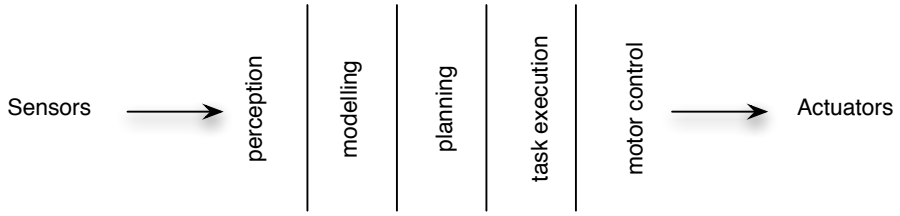

Figure 1. Brooks' (1986) characterisation of a "traditional" robot control architecture.

Brooks (1986) argued that a control system that was organised as a sequence of perception, modelling, planning, task execution and motor control is slow and introduces many unnecessary operations in the control cycle. He proposed that the connection between perception and action should be shorter and that the robot's architecture should be layered, as in Figure 2.

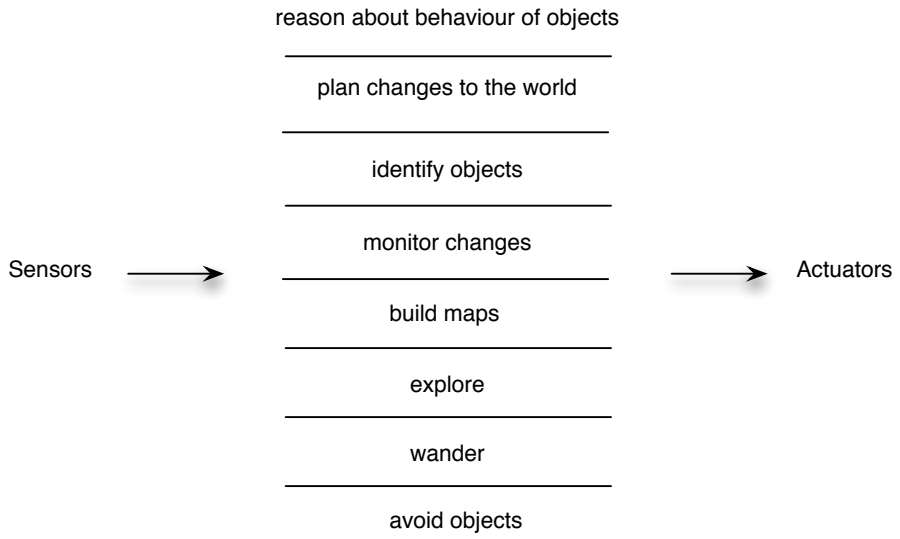

Figure 2. A layered subsumption architecture for a mobile robot [after (Brooks, 1986)].

In this configuration, the layers of the control system execute in parallel. The lowest layer uses a very simple reactive control system that is capable of responding rapidly to dynamic situations, such as avoiding obstacles. Progressively higher layers operate more slowly but are capable of

modifying the behaviour of lower layers lower, e.g., overriding obstacle avoidance if the goal is to push an object. Brooks also made a much stronger claim, that symbolic representations were not necessary and that they encouraged researchers to believe that general purpose solutions to many problems were possible, whereas their efforts would be better spent devising task-specific methods (Brooks, 1990; Brooks, 1991).

The bottom-up approach of allowing the task to dictate the requirements for the robot has been highly successful, as exemplified by the self-driving car. Our own experience in the international RoboCup robot soccer competitions (Sammut, 2010) has shown that, to achieve competent real-time performance, sensing, modelling and control must be specialised for the task. Unfortunately, this entails a great deal of work customising algorithms, devising special purpose behaviours and time-consuming fine tuning. Currently this is done by engineers. For a robot to have general intelligence, it should capable of learning specialised skills, and this is where representation and reasoning become important. Generality comes from the accumulation of many special skills rather than from a single skill that is general.

Brooks' strong position was, in a sense, a debating stance. His intention was to push the notion of intelligence without representation to its limits. It is now common to combine behaviour-based robotics with different kinds of representation and reasoning. This can be seen in various layered architectures for robots, to which we now turn.

3. Layered Architectures for Robots

Robot software architectures are usually layered (Nilsson, 2001; Albus, 1997; Gat, 1998; Ferrein, 2007), with the lowest layer consisting of reactive skills and progressively higher layers handling more deliberative tasks. In architectures like Gat's, shown in Figure 3, a deliberative layer is responsible for generating plans. The intermediate "sequencing" layer is responsible for choosing the plan that is appropriate for the current situation and the reactive layer executes the actions specified in the plan. The sequencer may change plans if the situation changes sufficiently.

Depending on the nature of the task, deliberation and learning may be too time consuming or too risky to do at run time. If tennis players think about their strokes during a rally or golf players are too conscious of their swing, they inevitably play badly. They must learn and practice these skills to automatise them so that they can execute them rapidly during a game. There may be some adjustment during a game to counter an opponent's play but the place for most planning and learning is during practice. So too, a robot should do most of the thinking it needs to do before it is deployed, so when it goes into operation, it does not need to deliberate. Thus, in Gat's model, the upper layer may be run "offline".

Robot soccer (Sammut, 2010) is a good example of the need for learning and deliberation prior to task execution and for reactive control during execution. A game can change so quickly that there is little time for a player to think about strategies. Instead, these are worked out in advance, just like a coach working with a team. Plans are made and practiced thoroughly before taking the field. In the laboratory, teams invent and program different behaviours and strategies and test them as well as they can. Often the behaviours are so specialised that they are only invoked in certain parts of the field or when team members and opponents are in certain positions. All behaviours are

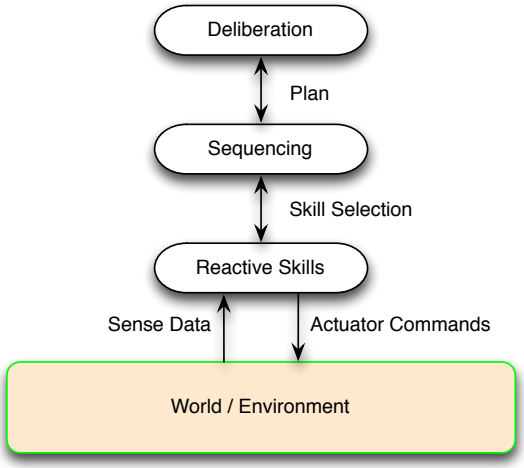

Figure 3. A three-layered architecture [after (Gat, 1998)].

reactive. Circumstances can change so rapidly and the outcomes of actions are so unpredictable that looking ahead more than one or two steps is futile. Rather than humans hand crafting skills, we would like the robot to be able to learn these for itself. Currently, some components of the soccer robot's software can be learned. For example, locomotion requires careful design and parameter tuning. Much of this time-consuming effort for this can avoided by learning the parameters for the gait during practice before the competition (Kim & Uther, 2003; Hester et al., 2010).

Different domains may have different time scales, some of which permit planning. For example, the real-time skills of the self-driving car must be reactive. However, route planning is not as time critical, so it can be done in advance and re-planned if changes are necessary, such as when unexpected traffic jams or blocked roads are encountered (Rogers et al., 1999). In some cases, extreme caution is required, so prior planning is extensive. For example, AI planning for space missions, such as the Mars rovers, has steadily increased (Estlin et al., 2007). However, much of this planning is done off-line in simulations because trying out a new plan onboard could cause an irrecoverable error. When a rover must traverse difficult terrain or extricate itself from sand, a twin rover is set up in a "Mars yard" on Earth to test a variety of strategies and nothing is executed on Mars until there is a high degree of confidence in chosen strategy.

4. Is Symbolic Representation Needed?

Brooks claimed that a robot did not need to model the world. However, almost all mobile robots do build models. One of the most important advances in mobile robotics has been the development of Simultaneous Localisation and Mapping (SLAM) (Dissanayake et al., 2001). These are methods that allow a robot to explore an unknown environment, make a map and localise itself within the map. SLAM is one of the innovations that made the self-driving car possible. The insight behind SLAM was that the world is inherently uncertain. Sensors are never entirely accurate. Motors never

behave precisely as commanded, and even if they do, external factors in the environment like the terrain or other agents make the outcome of an action unpredictable. These realisations gave rise to probabilistic robotics (Thrun et al., 2005).

In recent years, probabilistic approaches have dominated robotics and have pushed aside symbolic AI. However, the appearance that symbolic reasoning is not needed may be due to the fact that most applications that have been attempted so far have not needed it. When a powerful tool has been created, there is a temptation to seek out those problems for which the tool is well suited, avoiding those for which it is not.

As an example, let us consider a domestic robot. This is still more science fiction than reality, but why is creating a domestic robot so hard? There are hardware limitations. We would expect a robot to be capable of dextrous manipulation so that it can make the bed, clean the dishes, etc., but research in robot manipulation still has a long way to go before such things are possible. In addition, a domestic robot must be able to interact with the human occupants, which includes understanding commands and conversations. The challenge here is not just in speech recognition or gesture understanding. The robot must have a model of the home and models of what the human occupants expect the robot to do for them. It needs the models so that robot and human know they are referring to the same objects and so it can reason about commands or questions given to it. They are also needed when the robot is asked to explain its reasoning. Thus, symbolic representations are particularly important when a robot must work with humans and with other agents, not only in the home but in many applications where human-robot interaction is required.

Now that many low-level robotics problems have at least been partially solved, we are in a position to address problems where symbolic AI comes to the fore again.

5. The Role and Requirements of Cognitive Architectures in Robotics

As we have already seen, the main drawback of current robot software architectures is their inflexibility. Any significant change in the task or environment requires substantial re-engineering. However, a system capable of abstract representation and planning should be capable of adapting much more easily. For example, if the task-level specification of behaviours is expressed in a production rule language (Laird et al., 1987; Laird, 2008; Nilsson, 1994; Anderson & Lebiere, 1998) or logic (Levesque et al., 1997; Langley et al., 2009), then goals and behaviours are easy to change. More importantly, they are learnable.

Symbolic representations also make it possible for a robot to introspect, which is necessary for learning and also for human-robot interaction. As an example, consider a system like SOAR, which includes an episodic memory. Such a mechanism lets the robot learn from accumulated experience and also lets it engage in conversations about its past interactions with other agents, which can include other robots, software agents and humans. The fact that the representation is symbolic means that it can be mapped onto sentences in natural language.

Langley et al. (2009) describe the capabilities expected in a cognitive architecture. They briefly mention the need for dealing with uncertainty and being able to operate in dynamic environments. In robotics these are of paramount importance. Robots must operate in real time. The typical frame rate for a video camera is 25 or 30 frames per second. If the software cannot keep up with

the camera, important information may be missed, particularly when objects are moving quickly. Therefore, it must be possible to complete perception, modelling, decision making, execution and monitoring in the time it takes to grab one frame. Layered architectures achieve this speed by running the different layers at different time scales. The most time-critical code is confined to the lower levels and is kept as simple and efficient as possible. To be useful in real-time systems, a cognitive architecture must adopt a similar approach that allows several threads of computation to run concurrently, with reactive components being given the highest priority. Architectures that can compile plans into fast chunks should be amenable to such an implementation.

In addition to the requirements of real-time processing, the architecture must deal with uncertainty. It is tempting to assume that uncertainty can be confined to the lower layers of the architecture, which might use a probabilistic approach, allowing the higher layers to treat the world as if it were deterministic. Unfortunately, the effects of noise and other disturbances propagate upward. For example, sensor noise can lead to incorrect measurements, which can lead to the robot not knowing its location. It may be confident enough to believe it is in one of several possible locations or it may be completely lost, in which case it requires strategies for re-localising. In trying to find itself again, it will almost certainly interrupt its previous plan and must have a further recovery strategy.

Having some abstract knowledge of the capabilities of sensors and actuators allows a robot to compensate for shortcomings. For example, a rescue robot may use an infrared camera as its primary sensor for finding victims of a disaster. However, this may be deceived by other heat sources. A combination of thermal images with other visual, acoustics or chemical sensors often provides enough additional information to confirm recognition. Use of multiple sensors can be preprogrammed but, in unknown environments, novel combinations might be useful. The robot may also use its ability to change viewing angles or to manipulate objects to gain additional information. Thus, it is useful to have knowledge about the sources of uncertainty and to be able to reason about the capabilities of resources to determine how to minimise uncertainty.

6. Conclusion

In his classic introduction to mathematics, A. N. Whitehead (1911) claimed that:

> It is a profoundly erroneous truism . . . that we should cultivate the habit of thinking what we are doing. The precise opposite is the case. Civilisation advances by extending the number of important operations which we can perform without thinking about them.

That is, we become expert at something by learning it so well that we no longer have to think about it. So too, a robot becomes competent in a task when most of its actions can be determined with minimal processing. This does not mean the cognitive architectures have no benefit for robotics. Rather, the place for deliberation and learning lies in creating specialised skills in offline training, prior to actual missions. Cognitive architectures will find increasing use in robots as they become physically capable of more complex tasks, especially those requiring human interaction. To be useful, cognitive architectures must pay particular attention to being able to respond rapidly in dynamic environments where uncertainty is always present.

Acknowledgements

This work has been supported by the Australian Research Council Centre of Excellence for Autonomous Systems. Many of the opinions expressed here have been formed by participation in the RoboCup soccer and rescue robot competitions. Special thanks to the UNSW team members who have made many valuable contributions to this work.

References

Albus, J., McCain, H., & Lumia, R. (1989). *NASA/NBS standard reference model for telerobot control system architecture (NASREM)* (Technical Report NIST/TN-1235). National Institute of Standards and Technology, Gaithersburg, MD.

Albus, J. S. (1997). The NIST real-time control system (RCS): An approach to intelligent systems research. *Journal of Experimental and Theoretical Artificial Intelligence*, *9*, 157–174.

Anderson, J., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum.

Bonasso, R. P., , Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P., & Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, *9*, 237–256.

Brady, M. (1985). Artificial intelligence and robotics. *Artificial Intelligence*, *26*, 79–121.

Brooks, R. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, *6*, 3–15.

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, *47*, 139–159.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *RA-2*, 14–23.

DARPA (2007). http://archive.darpa.mil/grandchallenge/.

Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, *17*, 229–241.

Estlin, T., Gaines, D., Chouinard, C., Castano, R., Bornstein, B., Judd, M., Nesnas, I., & Anderson, R. (2007). Increased Mars rover autonomy using AI planning, scheduling and execution. *Proceedings of the International Conference on Robotics and Automation* (pp. 4911–4918). Rome.

Ferrein, A. (2007). *Robot controllers for highly dynamic environments with real-time constraints*. Doctoral dissertation, Knowledge-Based Systems Group, Rheinisch-Westfälischen Technischen Hochschule, Aachen, Germany.

Ferrein, A., & Lakemeyer, G. (2008). Logic-based robot control in highly dynamic domains. *Robotics and Autonomous Systems*, *56*, 980–991.

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence*, *3*, 251–288.

Gat, E. (1998). On three-layer architectures. In D. Kortenkamp, R. P. Bonnasso, & R. Murphy (Eds.), *Artificial intelligence and mobile robotics* (pp. 195–210). Menlo Park, CA: AAAI Press.

Hanford, S. D., Janrathitikarn, O., & Long, L. N. (2009). Control of mobile robots using the SOAR cognitive architecture. *Journal of Aerospace Computing, Information, and Communication*, *5*, 1–47.

Hester, T., Quinlan, M., & Stone, P. (2010). Generalized model learning for reinforcement learning on a humanoid robot. *Proceedings of the International Conference on Robotics and Automation* (pp. 2369–2374). Anchorage, AK.

Kim, M. S., & Uther, W. (2003). Automatic gait optimisation for quadruped robots. *Proceedings of the Australasian Conference on Robotics and Automation* (pp. 1–9). Brisbane: Australian Robotics and Automation Association.

Laird, J. E. (2008). Extending the Soar cognitive architecture. *Proceedings of the First Artificial General Intelligence Conference*. Memphis, TN.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1–64.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, *10*, 141–160.

Levesque, H. J., Reiter, R., Lespérance, Y., Lin, F., & Scherl, R. B. (1997). Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, *31*, 59–84.

Nilsson, N. J. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, *1*, 139–158.

Nilsson, N. J. (2001). Teleo-reactive programs and the triple-tower architecture. *Electronic Transactions on Artificial Intelligence*, *5*, 99–110.

Rogers, S., Fiechter, C.-N., & Langley, P. (1999). An adaptive interactive agent for route advice. *Proceedings of the Third International Conference on Autonomous Agents* (pp. 198–205). Seattle, WA: ACM Press.

Sammut, C. (2010). Robot soccer. *Wiley Interdisciplinary Reviews: Cognitive Science*, *1*, 824–833.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. Cambridge, MA: The MIT Press.

Thrun, S., & Urmson, C. (2011). Self-driving cars. Plenary talk presented at *the IEEE/RSJ International Conference on Intelligent Robots and Systems*. http://www.aiqus.com/questions/27762/sebastian-thrun-and-chris-urmson-on-self-driving-cars-at-iros-2011.

Velodyne (2012). http://velodynelidar.com.

Whitehead, A. (1911). *An introduction to mathematics*. New York: Henry Holt and Company.