Applied Mathematical Sciences, Vol. 6, 2012, no. 14, 675 - 688

Robot Self-Awareness: Exploration of Internal States

Anna Gorbenko

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
gorbenko.ann@gmail.com

Vladimir Popov

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
Vladimir.Popov@usu.ru

Andrey Sheka

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
Andrey.Sheka@gmail.com

Abstract

A self-aware system has the possibility of dealing with novel situations more effectively than a system without self-awareness. A self-aware system can attend to its own internal states, thus providing a means of generating introspection and self-modification capabilities. A robot needs a capability to attend to its internal states in order to be genuine self-aware. Internal states can be made up of emotion, belief, desire, intention and expectation or it can be processes such as sensation, perception, conception, simulation, action, planning and thought. It is crucially important to be aware of its own emotions, perceptions, beliefs and intentions during the recognition process. Currently, developments in the field of self-awareness of robots are mainly based on a mimicry of human internal states. It is difficult for systems developers to specify specific internal states for all possible conditions and situations. It is obvious that such systems have very limited opportunities

for self-development. In this paper we consider an approach that allows the robot to generate their own internal states. These internal states are not similar to the human internal states. Such property gives the system of internal states plenty room for self-development. We propose a new model of genetic algorithm for analysis of robot control system and generation of new internal states.

Mathematics Subject Classification: 68T40

Keywords: self-awareness, genetic programming, genetic algorithms

1 Introduction

Current AI systems, including robotic systems, are unacceptably brittle in the face of change. System designers and developers must explicitly specify all actions and behaviors in order for the system to work as intended such that it achieves its design goals. In open complex and dynamic environments it becomes difficult for systems developers to specify specific actions and behaviors for all possible conditions and situations. Software development approaches to structured and well defined problems do not typically scale in complex and dynamic environments because there is a huge, possibly infinite, number of possibilities that need to be catered for; it is unreasonable to expect that system developers can foresee and develop appropriate responses for all relevant eventualities. Once deployed, systems are effectively limited by a static set of instructions that encode their designers understanding, conception and perception of the domain in the form of action and behaviors. As a consequence, current systems are limited to domain specific applications where they can perform under a small and finite set of conditions that have been specifically anticipated and represented in a given application. It is not surprising that they fail to perform in open complex and dynamic environments.

Even if a robot has no true self-awareness it can have some characteristics of self-awareness, such as having emotional states or the ability to recognize itself in the mirror. Such characteristics can significantly improve efficiency of a robot in open complex and dynamic environments.

A self-aware system has the possibility of dealing with novel situations more effectively than a system without self-awareness, because it would have the capacity for introspection that would allow it to inspect and exploit its representations, e.g. internal state. A self-aware system can attend to its own internal states, thus providing a means of generating introspection and self-modification capabilities. A robot needs a capability to attend to its internal states in order to be genuine self-aware (e.g. [1]). Internal states can be made up of emotion, belief, desire, intention and expectation or it can be processes

such as sensation, perception, conception, simulation, action, planning and thought. It is crucially important to be aware of its own emotions, perceptions, beliefs and intentions during the recognition process. Currently, developments in the field of self-awareness of robots are mainly based on a mimicry of human internal states (e.g. [1], [2]). It is difficult for systems developers to specify specific internal states for all possible conditions and situations. It is obvious that such systems have very limited opportunities for self-development. In this paper we consider an approach that allows the robot to generate their own internal states. These internal states are not similar to the human internal states. Such property gives the system of internal states plenty room for self-development. We propose a new model of genetic algorithm for analysis of robot control system and generation of new internal states.

2 Self-Awareness

Robotic systems have been developed to be aware of their own motion [3], [4], able to imitate [5], [6], driven by emotion [2], and able to change their own models of their physical embodiment [7]. Important work in cognitive robotics in reasoning about action and reasoning about knowledge [8], [9], [10] is relevant. In [3] developed an infant-like humanoid robot called NICO that can recognize its own motion in its visual field, including in a mirror. NICO expects to see motion in its visual field whenever certain motor movements commence after a certain time. It learns this time characteristic through experimentation. It labels motions that appear in the visual field within this learned time frame as its own motion, thus it can distinguish itself from others based on the idea of linking motion to time. In [5] considered an approach to consciousness is to maintain consistency of cognition and behavior of self and others in order to understand the behavior of self and others. When a system reaches a state where this behavior of self and others is understood, the system is deemed to be conscious. They determine that the imitative behavior is adequate for analyzing consistency in cognition and behavior of self and others. They conducted four experiments using a robot's imitating actions, including its own actions in a mirror. The result is that the robot passed a mirror test with 70% accuracy [6]. Being conscious in this sense, the robot was able to discriminate itself and others much of the time, however the relationship between consciousness and awareness in this scenario or context is not discussed or clarified. In [2] developed the Intelligent Soft-Arm Control (ISAC) robot that is not self-aware in the sense that it cannot recognize itself in a mirror, but it can deliberate on its emotions based on memory experience. Self-reflection, self-awareness and sense-of-self are represented by a self-agent which consists of a set of agents interacting with memory systems. The emotion that emerges from an activity of experience is learned and stored in memory systems. When

an event occurs, emotions activate the episodic memory which in turn activates cognitive control to suppress current behavior and execute required behavior. Robots have been developed that exhibit an adaptation capability for their own body [7]. These robots can recover from damage or failure that occurs to their body. A robot continuously creates a concept of its own physical structure (self-modeling) and uses this self-model to generate forward locomotion with four legs initially without knowing what its body actually looks like. When the robot's structure changes unexpectedly, it can reform its internal self-model to generate new behaviors to compensate and accommodate these changes. In this case, it remodels the concept of its own physical structure to generate forward locomotion with three legs when one of its legs is removed. This is possible because it has a model of its own physical structure. In [1] explored robot self-awareness and the role that attention plays in the achievement self-awareness and proposed a new attention based approach to self-awareness. In particular, in [1] provided a new self-modifying framework for developing an attentive robot with self-awareness based on an architecture that supports the ability of a system to focus attention on the representation of internal states.

Many of current approaches do not focus on directing a robot's attention to its own internal processes. If we add an attention process to a robot so that it can focus on processes that happen internally during self-recognition activities, then we would consider it to be self-aware. What is crucially important is not the ability to recognize itself in a mirror, but rather to be aware of its own internal states. If a robot has totally lost all of its outward facing sensations, it may not be aware of its environment, however it can still be aware of itself.

Following [1] we define self-awareness to be the capability of an agent to focus attention on the representation of internal states. However, in contrast to [1] we are not trying to describe any specific internal states, such as emotion, belief, desire, intention, expectation, sensation, perception, conception, simulation, action, planning, thought, etc. Also, we do not try to interpret the semantics of internal states of a robot in some natural language. Since self-awareness needed to improve efficiency of a robot in open complex and dynamic environments, a capability to understand the sense of internal states of a robot is not essential to achieve the goal. Moreover the need to understand internal states of a robot makes the problem immeasurably more difficult. For example, we know that children can learn to walk. We even know how to teach them to walk. But we can not teach a robot in the same way, because we do not fully understand the proper sequence of internal states of a child.

Regardless of an approach to definition of internal states it is clear that internal states are some states of the robot control system and computing resources or changes of these states. Therefore, as internal states we consider precisely states of the robot control system and computing resources and changes of these states.

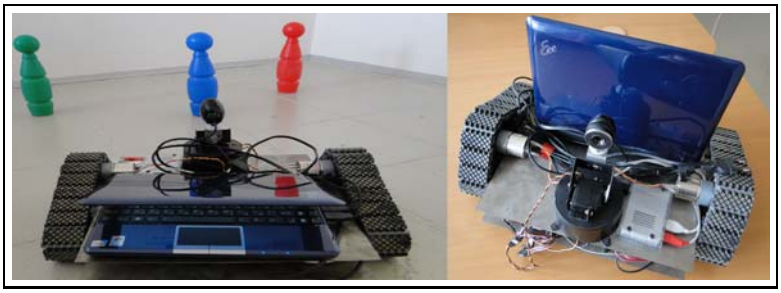

Figure 1: Robot Kuzma-II.

3 Mobile Robot Testbed

We have been implementing our approach and framework on the robot Kuzma-II (see Figure 1). Design of this robot based on the well-known Johnny 5 Robot [11]. By utilizing heavy duty polypropylene and rubber tracks with durable ABS molded sprockets the robot has excellent traction. It includes two 12vdc 50:1 gear head motors and the Sabertooth 2 x 5 R/C motor controller. The body of Johnny 5 Robot wasn't used. The original panels of Johnny 5 Robot were replaced with panels of a large area (30×33 cm) of stainless steel. This allowed us to significantly increase the payload of the robot. Also, it provided a much greater structural strength. The robot is equipped with one or more batteries AcmePower UC-5. Also used a standard laptop battery. The electronic system based on SSC-32 microcontroller. Onboard computer of this robot is Asus Eee PC 1000HE with OS Windows XP SP2. The robot is equipped with a 2 DOF robotic camera (USB web camera Live! Cam Video IM Pro (VF0410)). Using a wireless connection robot has access to resources of a cluster. We use heterogeneous cluster based on three clusters (Cluster USU, Linux, 8 calculation nodes, Intel Pentium IV 2.40GHz processors; umt, Linux, 1664 calculation nodes, Xeon 3.00GHz processors; um64, Linux, 128 calculation nodes, AMD Opteron 2.6GHz bi-processors) [12].

The basic robot control system developed in Java. This system is designed to work with devices. Intelligent functions assigned to the advanced robot control system. This system developed using the C# programming language on the .NET 2.0 framework. The robot uses a visual navigation system. The advanced robot control system can automatically generate recognition modules for specific tasks and specific environments. The only criterion that determines the need for generating new module is the quality of recognition. If current module provides detection of low quality, then the system generates a new module based on neural networks. If current module provides detection of high quality, then the system generates a new module based on simpler neural networks or threshold schemes.

Gridworlds are popular testbeds for planning with incomplete information.

In [13] studied a fundamental planning problem, localization, to investigate whether gridworlds make good testbeds for planning with incomplete information. In [13] found empirically that greedy planning methods that interleave planning and plan execution can localize robots very quickly on random gridworlds or mazes. Thus, they may not provide adequately challenging testbeds. On the other hand, in [13] showed that finding localization plans that are within a log factor of optimal is **NP**-hard. Thus there are instances of gridworlds on which all greedy planning methods perform very poorly. In [13] showed how to construct them. These theoretical results help empirical researchers to select appropriate planning methods for planning with incomplete information as well as testbeds to demonstrate them. In our investigations we use the localization problem as the primary objective of our robot. We use hard instances of this problem. Our gridworlds marked with colored skittles (see Figure 1). Accordingly, the robot uses skittles as landmarks. We use different sets of colors for stimulation of generation of new recognition modules.

Kuzma-II has a robust design. It is suitable for many hours of experiments in autonomous mode. The robot control system capable to rapid self-modification. It is clear that if we observe changes of the robot control system and its performance, then we find the dynamically changing world which provides us well testbed for our investigations.

4 Exons and Introns

In this section we consider biological observation which used for chromosome representations of individuals for genetic algorithms used to solve the previously considered problem. DNA and proteins are polymers, constructed of subunits known as nucleotides and amino acids, respectively. The sequence of each protein is a function of a DNA sequence which serves as the gene for that protein. The cellular expression of proteins proceeds by the creation of a message copy from the DNA template into a closely related molecule known as RNA. This RNA is then translated into a protein. One of the most unexpected findings in molecular biology is that large pieces of the RNA are removed before it is translated further. In the 1960s non-bacterial ribosomal RNAs were found to be synthesized as a long precursor RNA which was subsequently processed by the removal of apparently functionless internal "spacer" sequences. In the 1960s a similar processing was found to apply to eukaryotic precursor messenger RNAs [14]. In the mid 1970s it was found that the some of the internal sequences interrupted the protein-encoding part of the corresponding mRNAs. The majority of eukaryotic genes display a complex structure in which code for protein are interrupted by intervening, non-coding sequences. Initial transcription of these genes results in a pre-message RNA molecule from which segments must be accurately removed to produce a translatable message. The

retained sequences are known as exons (see [15]), while the removed sequences are known as introns. Therefore, we can consider a complex structure in which code are interrupted by intervening, non-coding sequences.

5 Genetic Programming Model of Exploration of Internal States

The main idea of genetic programming consists in the adaptation of biological evolutionary models for use in software systems. The basic operating element in the genetic programming is a chromosome that is responsible for the configuration of a system. Chromosome consists of genes, each of which is responsible for a certain parameter of the system. Over a set of chromosomes an evolutionary mechanism operates providing variation, heritability and natural selection. To ensure the heritability in the population used crossover operator which creates a new chromosome by using the genes of two parental chromosomes. Variability is achieved through the use of mutation operator, which in some way alter the genes of chromosomes. Natural selection in populations is based on a fitness function of chromosome.

In models of artificial evolution is possible to use models of genes that differ from the biological gene that gives significant advantages for some problems. In the simplest case, we introduce genes as a sequence of bits that encodes the state of the system. This model is the most convenient in terms of data presentation. However, it is not always convenient in terms of application use. More complex model proposes to consider genes as a sequence of abstractions. It may be, for example, graphs, strings, sequences of real numbers. Such model allows a gene to provide information about the system in its original form, without intermediate transformation. More complex model of the gene allows us to allocate the introns and exons.

According to our biological model of genetic programming exons are responsible for direct configuration of the system, introns contain meta information about ongoing evolutionary processes. Note that exons in its configuration does not differ from genes in the previous model. In view of the fact that introns contain information obtained as a result of evolutionary changes in the gene, they must be closely related to evolutionary mechanisms. Therefore, it is necessary to consider the function of an intron jointly with operators of crossover and mutation. Note that in order to upgrade of the approach we can also consider some functions of introns which linked to other parts of the evolutionary mechanism.

When analyzing the functions of introns in the biological model, we identified the following roles of introns for applications:

1. Connectivity of genes. In the process of evolution there are situations

when certain genes show good results in aggregate. However, individual genes do not affect positive on properties of the system. Therefore appropriate to use introns to associate genes. Due to this evolutionary operators receive information about features of genes. This allows them efficiently move, store, and change properties of the system.

2. Statistical container. With the help of introns, we are able to save statistical information on the gene. Using this information we can stimulate evolutionary processes.

3. Revision history. During the evolution genes changed. These changes itself are useful. If we know changes that have occurred we have possibility to build their strategy based on the evolution. If we need to maximally preserve inheritance, then we can recover genes from history. If we pursue the goal of maximum variability in generations, then we need to avoid overlaps with genes from history.

From the point of storage, introns is conveniently considered as a set of objects. Evolutionary operators respond only to those objects from the set with which they can work, but others simply ignore it. Thereby possibly changing the set of operators directly during the evolution which makes it possibility to guide an evolution in the right direction.

From the point of representation of software package in the evolutionary model chromosomes is a system configuration. Class of agents is a set of agents that meet a certain standard input and output data. Type of gene is an agent with the set of possible options. Instance of the gene is an agent with certain parameters. Thus each generation of evolution corresponds to some change of robot control system. In our case we use the generator of recognition modules for modification of robot control system. The results of the genetic algorithm we use as an external stimulus to change one or more recognition modules.

Note that a crossover operator can be arranged over instances of a gene of one type only in the presence of two or more independent parameters of the agent. Generally, mutation operator replaces current copy of a gene by other gene of the same type. Genes encode the information that applies to each of three subsystems of the agent. Since subsystem is defined at the time of the designing of an agent, information that can be stored in a gene is a collection of some of constants that are responsible for configuring of subsystems. Operators of mutation and crossover for predefined systems are defined in a natural way. Employed subsystems are agents that can be activated by the parent agent from some class of agents. Note that over time the number of agents which belong to the same class can be changed. For the organization of correct operations of evolutionary mechanism employed subsystems must promptly update an information on the set of agents which are included in

classes. In addition, we need to adjust crossover and mutation operators so that they can work correctly with all agents of a given class. Note that for employed subsystems crossover and mutation operators constructed on the basis of operators which used at work with sets. Let $f(x)$ be a function which is responsible for the quality of solutions. Let $g(x)$ be a function which is responsible for the used computational resources. To deployment of an evolutionary mechanism it is necessary to have an unified fitness function. Of course, we can consider as a fitness function, for example, $f(x)g(x)$ or $f(x)(g(x)+f(x))$. However, in this case the fitness function will have a misty sense. Our experiments showed that the best results are achieved when using $f(x)$ as the fitness function. With such fitness function values of $g(x)$ are encoded in introns.

6 Computational Experiments

For our computational experiments, we chose the population size, the number of classes and the variety of agents in each class to be 10, 6, and 10, respectively. For computational experiments, we developed a system for prediction of performance. Work of our genetic algorithm continues until there was an increase of values of the fitness function. In each experiment, genetic algorithms find the best possible configuration of the system at 30 generations.

We compared results of genetic algorithm for real measurements of performance (see Figure 2) and for data which obtained from the system for prediction of performance (see Figure 3). Our experiments showed that results for real measurements of performance and for data which obtained from the system for prediction of performance differ insignificantly. In particular, results of the genetic algorithm, which started from population which obtained using the system for prediction of performance, for real measurements of performance is presented in Figure 4.

7 The c-Fragment Longest Common Subsequence Problem

As mentioned above, the best results are achieved when using $f(x)$ as the fitness function. In this case values of $g(x)$ are encoded in introns. Since $g(x)$ is a function which is responsible for the used computational resources, it is natural to consider $g(x)$ as some characteristic of internal states of the robot. The robot can demonstrate the same performance, but at the same time it can be in a substantially different internal states. To analyze internal states of the robot and to investigate the dynamics of their changes, we introduce the following new combinatorial problem.

Generation	Chromosome	Agent	Fitness
1	1	558287	3.1269594118791555E-10
	2	558234	1.4416066826577117E-9
	3	559284	1.5719790987161234E-8
	4	558984	2.0802415095987675E-9
	5	058284	2.8002600036883247E-9
	6	725151	6.7683280447743205E-9
	7	725157	1.4562709159476911E-8
	8	725151	3.639743102798477E-8
	9	725150	5.961161755534219E-8
	10	727151	1.7926351339250583E-8
20	1	871663	1.2591806364297917E-5
	2	871668	7.73848123797181E-5
	3	871620	7.642266973803008E-6
	4	871690	1.7081796667793985E-5
	5	971660	7.628818116329376E-5
	6	878660	2.1604965555893806E-6
	7	872360	1.2006609272865991E-4
	8	172660	3.9118136227531894E-5
	9	872660	7.965441672272984E-5
	10	872660	1.419790015634894E-4
30	1	802362	9.664436419503133E-5
	2	807360	9.115564906047902E-5
	3	802360	1.884639449129221E-4
	4	802360	1.6265194628480197E-4
	5	802360	1.0404586494822476E-4
	6	902360	6.138852814213215E-5
	7	802300	6.151080437663375E-5
	8	802360	1.3983212098514318E-4
	9	802360	1.0835102666928983E-4
	10	802360	9.562119409621382E-5

Figure 2: Results of genetic algorithm for real measurements of performance.

Given two sequences S and T over some fixed alphabet Σ, the sequence T is a subsequence of S if T can be obtained from S by deleting some letters from S. Notice that the order of the remaining letters of S bases must be preserved. The length of a sequence S is the number of letters in it and is denoted as $|S|$. Given two sequences S_1 and S_2 (over some fixed alphabet Σ), the classic longest common subsequence problem (LCS) asks for a longest sequence T that is a subsequence of both S_1 and S_2. We consider the c-fragment longest common subsequence problem (see e.g. [16]) which is a special case of LCS:

Generation	Chromosome	Agent	Fitness
1	1	906124	8.352839058261439E-7
	2	906124	8.352839058261439E-7
	3	936124	4.089569606797197E-7
	4	906184	1.0741894471864074E-6
	5	936124	4.089569606797197E-7
	6	394040	1.529745975894662E-7
	7	394045	8.315983586087238E-8
	8	394040	1.529745975894662E-7
	9	394040	1.529745975894662E-7
	10	394040	1.529745975894662E-7
20	1	801562	1.539983774271137E-4
	2	801562	1.539983774271137E-4
	3	801582	1.7176128803758332E-5
	4	801566	6.728819044756761E-5
	5	801542	2.6428726599364868E-5
	6	802562	1.3172984666916108E-4
	7	802562	1.3172984666916108E-4
	8	802562	1.3172984666916108E-4
	9	802562	1.3172984666916108E-4
	10	803562	1.0401028041069673E-4
30	1	801562	1.539983774271137E-4
	2	801532	1.603822352629095E-5
	3	804562	9.755845680833868E-5
	4	001562	3.8033472907018084E-5
	5	801542	2.6428726599364868E-5
	6	801562	1.539983774271137E-4
	7	801562	1.539983774271137E-4
	8	801462	9.79603863101633E-5
	9	801569	1.067925128587828E-4
	10	801563	1.7409211251253976E-5

Figure 3: Results of genetic algorithm for data which obtained from the system for prediction of performance.

FLCS:

INSTANCE: *An alphabet* Σ*, sequences* S_1 *and* S_2*,* $S_1, S_2 \in \Sigma^*$*, where* S_1 *and* S_2 *are divided into fragments of lengths exactly* c *(the last fragment can have a length less than* c*).*

TASK: *Find a longest common subsequence* T *of* S_1 *and* S_2 *such that the allowed matches are those between fragments at the same location.*

Note that the longest common subsequence is a classical distance mea-

Generation	Chromosome	Agent	Fitness
1	1	801562	9.986022804193584E-5
	2	801522	9.448180495309388E-6
	3	801562	1.0758946073448435E-4
	4	805562	7.655306692847537E-5
	5	801562	1.0638539095731142E-4
	6	801562	1.0870067611647092E-4
	7	804562	6.597716897967614E-5
	8	801562	1.1417571985732377E-4
	9	801562	1.0826140079050429E-4
	10	801522	9.227518633880476E-6
20	1	801662	1.1181214119124201E-4
	2	831662	5.046903454959181E-5
	3	801662	1.1821804511365691E-4
	4	801662	7.747813282374215E-5
	5	801672	7.433738955230544E-5
	6	801662	7.638092426861063E-5
	7	801662	1.1487742553431696E-4
	8	811662	4.8893036598293416E-6
	9	801262	6.0732268759693924E-5
	10	801662	1.1587933458492471E-4
30	1	802662	9.283647319991188E-5
	2	808662	1.8277743017402426E-6
	3	301662	3.1582167303904335E-6
	4	801662	1.0871058708515887E-4
	5	801642	1.7640621053528226E-5
	6	861662	3.983929780367871E-5
	7	801662	8.612628488280334E-5
	8	851662	1.621997644500425E-5
	9	401662	5.281334905991253E-5
	10	801662	7.391434896658494E-5

Figure 4: Results of the genetic algorithm, which started from population which obtained using the system for prediction of performance, for real measurements of performance.

sure for strings. Our special case of LCS needed for accurate comparisons of parts of introns. For the initial comparison it is sufficient to consider FLCS. However, for our experiments, more accurate information is provided by consideration of FLCS for k subsequences. In our investigations and experiments, to analyze internal states of the robot, we consider only similarity of introns. However, plenty of room for further investigations reserves consideration of

dissimilarity of introns. For instance, it is interesting to consider such problems as the longest common non-supersequences problem, the shortest common non-subsequence problem, etc [17].

8 Conclusion

In this paper we have just outlined a framework for further research in the field of self-awareness on the basis of machine-oriented internal states. In particular, we proposed a new model of genetic algorithms which based on the use of exons and introns. These genetic algorithms allow to investigate internal states of a robot. Experiments discussed in this paper showed that such genetic algorithms can be used to generate new internal states. In particular, our approach allows the robot to generate their own internal states. These internal states are not similar to the human internal states. Such property gives the system of internal states plenty room for self-development. We propose a new model of genetic algorithm for analysis of robot control system and generation of new internal states.

ACKNOWLEDGEMENTS. The work was partially supported by Analytical Departmental Program "Developing the scientific potential of high school" 2.1.1/14055.

References

[1] R. Novianto and M.-A. Williams, The Role of Attention in Robot Self-Awareness, *The 18th IEEE International Symposium on Robot and Human Interactive Communication*, (2009), 1047-1053.

[2] K. Kawamura, W. Dodd, P. Ratnaswasd, and R.A. Gutierrez, Development of a robot with a sense of self, *Proceedings of the 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, (2005), 211-217.

[3] P. Michel, K. Gold, and B. Scassellati, Motion-Based Robotic Self-Recognition, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2004), 2763-2768.

[4] A. Gorbenko, V. Popov, and A. Sheka, Robot Self-Awareness: Temporal Relation Based Data Mining, *Engineering Letters*, 19 (2011), 169-178.

[5] T. Suzuki, K. Inaba, and J. Takeno, Conscious robot that distinguishes between self and others and implements imitation behavior, *Proceedings of the 18th international conference on Innovations in Applied Artificial Intelligence*, (2005), 101-110.

[6] J. Takeno, K. Inaba, and T. Suzuki, Experiments and examination of mirror image cognition using a small robot, *Proceedings of the 2005 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, (2005), 493-498.

[7] J. Bongard, V. Zykov, and H. Lipson, Resilient machines through continuous self-modeling, *Science*, 314 (2006), 1118-1121.

[8] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi, *Reasoning about knowledge*, MIT Press, Cambridge, 2003.

[9] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R.B. Scherl, GOLOG: A logic programming language for dynamic domains, *The Journal of Logic Programming*, 31 (1997), 59-83.

[10] R. Reiter, *Knowledge in action: logical foundations for specifying and implementing dynamical systems*, MIT Press, Cambridge, 2001.

[11] http://www.lynxmotion.com/c-103-johnny-5.aspx

[12] http://parallel.imm.uran.ru/mvc_now/hardware/supercomp.htm

[13] C. Tovey, and S. Koenig, Gridworlds as Testbeds for Planning with Incomplete Information, *Proceedings of the AAAI Conference on Artificial Intelligence*, (2000), 819-824.

[14] K. Scherrer, G. Spohr, N. Granboulan, C. Morel, J. Grosclaude, and C. Chezzi, Nuclear and cytoplasmic messenger-like RNA and their relation to the active messenger RNA in polyribosomes of HeLa cells, *Cold Spring Harbor Symposia Quantitative Biology*, 35 (1970) 539-554.

[15] W. Gilbert, Why genes in pieces? *Nature*, 271 (1978), 501.

[16] V. Popov, Arc-preserving subsequences of arc-annotated sequences, *Acta Universitatis Sapientiae. Informatica*, 3 (2011), 35-47.

[17] L. Zhang, On the approximation of longest common nonsupersequences and shortest common nonsubsequences, *Theoretical Computer Science*, 143 (1995), 353-362.

Received: September, 2011