A New Approach to Robot's Imitation of Behaviors by Decomposition of Multiple-Valued Relations

Uland Wong and Marek Perkowski
Department of Electrical and Computer Engineering,
Portland State University
P.O.Box 751, Portland, Oregon, 97207-0751

Abstract

Relation decomposition has been used for FPGA mapping, layout optimization, and data mining. Decision trees are very popular in data mining and robotics. We present relation decomposition as a new general-purpose machine learning method which generalizes the methods of inducing decision trees, decision diagrams and other structures. Relation decomposition can be used in robotics also in place of classical learning methods such as Reinforcement Learning or Artificial Neural Networks. This paper presents an approach to imitation learning based on decomposition. A Head/Hand robot learns simple behaviors using features extracted from computer vision, speech recognition and sensors.

1. Building a Learning and Imitating Robot Puppet with High School Students

High school robotics is advocated by many authorities as the best way to enthuse young people to become interested in mechanical design, mathematics, programming, electronics and mechanics [3,5]. In our high school robotics outreach program at PSU we want to appeal also to those youngsters who have interest in sculpture, artistic design, theatre and puppets. For the last three years the second author works with high school students on a variety of long-term robot projects. We integrate mechanical design of robots, interface design, logic synthesis, computer vision and machine learning. The ultimate technical goal of this project is to build a theater of fully autonomous robotic creatures that will live semi-independent life in our Laboratory, communicating with humans and among themselves using speech and natural language and gestures seen by their eye-cameras [16]. The didactic goal, however, is to work with several student groups: talented, high risk, disabled and female students and help them to find creativity in themselves. The robots will be equipped with rule-based general behaviors such as: language generation and parsing, "robot morality and etiquette" and knowledge usage. They are being taught rather than programmed. Example of a puppet of our Portland International Cyber Theatre is shown in Figure 1. It is called BUG, or Big Ugly Greeter. The robot was designed by a small group of high school students, guided by the first author of this paper. Most of the design was done by him in the summer of 2001 and is being programmed since then. The software will integrate Visual Basic, Visual C++, Lisp and much publicly available software for speech synthesis, speech recognition, image processing and computer interfacing [19,20,22,23,24]. We use also software developed previously at PSU, U.C. Berkeley, and Technical University of Freiberg for machine learning [6,6,9,10,11,21] and image processing [4]. In our system the only type of actuators are inexpensive servo-motors that can be purchased in most hobby shops or through Internet. The cost of such servo is about $ 8.50 when purchased in larger quantities, which allows to build inexpensive robots with many degrees of freedom. The hand/head robot is controlled by a total of 21 servos. Four fingers have 3 degrees of freedom each and the hand has a total of 16 servos. Head has 5 servos. There are 4 touch sensors on tips of each finger and one motion sensor for the head. In addition, Intel cameras and microphones serve as the robot's inputs. There will be also several additional sensors on hand and face: temperature, touch, infrared, and other. The signals from sensors are either digital or analog and converted in A/D converters to multi-valued input signals of the software-realized Reactive State Machines (RSM).

2. Learning Behaviors as Inducing Relations from Data

Many problems can be described using tabular form in which rows correspond to objects (elementary behaviors, minterms) and columns to attributes (features, variables) corresponding to objects [1,10,14,17]. For instance, object *behavior_1* has input variable values *COLOR_OF_HAIR = black* and *HAND_MOVEMENT = horizontal* and output variable THREE_*FINGERS_UP = 1*. This means, that when camera sees a person with *black hair* who is waving his hand *horizontally* and robot hand is raising *three fingers up* then this behavior should become an instance of a learned, generalized robot's behavior. Or, in other words, this behavior is reinforced by the learning system and will be used,

together with other positive and negative objects to induce a general formula for robot's behaviors. Such tables, object/attribute tables are the inputs to many types of learning algorithms: Rough Set, Decision Trees, Neural Nets, Reinforcement Learning, DNF minimization and Functional Decomposition [13,19]. In general, the variables are multiple-valued. Symbolic variables are encoded and the encoding can be for both nominal and non-nominal variables. For instance, a ternary variable *COLOR_OF_HAIR* encodes values: *bald = 0, black = 1, yellow = 2*. The ***generalized don't cares*** are allowed in our system, which means instead of a single value of attribute for an object, some subset of possible values of this attribute is allowed. For instance, color of hair is *black* or *brown* but **not** *yellow* in an input variable *COLOR_OF_HAIR*, or the robot will show *one* or *two* or *three* fingers up but **not** *four* as a value of the output variable *FINGERS_UP*.

Behavior of a robot is described in terms of a hierarchy of (Mealy) finite state machines in which every machine is composed of a multiple-valued combinational function and a memory. The inputs to the combinational function are primary inputs from sensors (including vision and speech) or outputs from other state machines and the current state signals of this state machine. The outputs of the state machine are: the next state signals of this state machine, and the outputs of this state machine which either go to other state machines or to the actuators. Counters are used for timed behaviors. We called these machines Reactive State Machines, as they react to events of the environment with robot's behaviors.

There are several machine learning methods that can be used to induce combinational modules. We used the method of decomposition of multiple-valued relations [1,14,17,21] but here we will specifically concentrate on the method for bi-decomposition of multiple-valued relations from [11,21] which is now a part of MVSIS tools from University of California in Berkeley [25]. Actually, any tool included into MVSIS can be used, but we experiment now only with bi-decomposition. The input data are tables of MV relations (with very high percent of don't cares, called don't knows in Machine Learning) and the output is a multi-level netlist of multi-valued gates described by gates such as MINIMUM, MAXIMUM, MODULO-ADDITION and so on. This netlist can be interpreted as a multi-input, multi-output control program of a robot, describing a set of generalized interrelated behaviors, similar but much more complex than the *behavior_1* above**.** Synthesized behaviors are in a form of BLIF-like multi-valued logic format so that they can be edited manually (learning by brain surgery) by students. They can be also translated to a C program whose subroutine calls are interpreted as calling executions of certain robot actions.

3. Hierarchy of Reactive State Machines for Robots

The robot performs simple programmable movements: MOTOR-1-UP, MOTOR-2-DOWN, MOTOR-3-LEFT, PICK, RELEASE. They correspond to states of output variables and can be timed or cyclic, for instance leg movements for certain gaits. For instance, each finger has 3 servo-motors, each servo has a specified number of states which correspond to angles of finger's joints. Variables for head correspond for instance to angles of head rotations. All signals, input, output and internal, are programmed as multi-valued. Timing information is added. For instance: MOTOR-1-UP for 2 seconds, MOTOR-2-LEFT and MOTOR-3-UP for 3 seconds. One multi-valued variable is used for time, with 4 bits. It describes time for both input and output variables. This allows to give commands such as: ***IF*** INPUT-SIGNAL-SENSOR-1 for two seconds **AND** INPUT-SIGNAL-SENSOR-3 for one second, **THEN** generate sequence MOTOR-1-UP and MOTOR-2-LEFT and MOTOR-3-UP for 10 seconds. State machines are hierarchical and distributed, each component specifies certain behavior, for instance, it includes counters for counting time. Even for a simple learned machine, the Cartesian products of parallel machines create extremely large spaces of possible observable behaviors, which can make an impression of intelligence and free choice. The user describes the learning data in form of tables available in a new window of software. The user cannot change the hierarchy of machines and the general decomposition to modules, but can teach new functionalities of each module or change essential to vacuous variables or states and vice versa.

There exists also a voice recognition module (excellent toolbox from Microsoft). The human can control the robot with simple voice commands such as LEFT, RIGHT, STOP. Image processing subsystem allows to recognize rough shapes and colors. For instance, the color can be used to recognize the human, the robot has not trouble to recognize Uland's hair (Figure 6). Big area of green (a jacket, etc.) symbolizes man Green, red will is man Red, etc. The experiments illustrate the behavior-based programming based on Reactive State Machines (RSMs) [2,5,8,13,15,16,18,19]. Using this paradigm, the robot programs (i.e. RSMs) are constructed as a hierarchy of behavior modules that execute concurrently in a multi-tasking environment. This enables a robot builder to create robots that

exhibit simple behaviors to which more sophisticated behaviors can be added simply by coding **higher level** behavior modules. This way of organizing a hierarchy of behavior modules is also termed ***subsumption architecture*** [2]. This proved to be a unique and concrete way of introducing multitasking, and issues relating to real-time control in the computer science curriculum and is used at MIT, CMU etc. What is new to our approach, is the acquisition of the machines' functionalities in terms of: ***examples being generalized don't cares, and learning by Decomposition of Relations.*** Typically, the tasks only provide an abstract description of the desired mapping from perception to action. Much of the detailed specifications about how to behave in a particular situation is not given and may not be known. For example, a typical robotics task is to navigate (manipulate) from a starting point to a goal point. It can be learned by examples rather than being programmed in detail in software language.

4. Modes of Robot Learning

The "brain" of the robot is built not only by writing software but by teaching the robot on examples. This is a standard "learning from supervisor" approach, and the student is the supervisor. He creates all sequences for training the reactive state machine. It is as the parent would teach the child by ***re-wiring directly his brain*** based on positive and negative examples. The set of sequences is incomplete, so the machine performs the ***generalization***, automatically. Adding or removing new rules, by the human supervisor or automatically/randomly, will change the behavior. The students can experiment with this a lot. Due to the open ended nature of robotics problems, a lengthy process of trial and error is often necessary to answer typical questions of robot behavior clearly enough to develop a working algorithm. Students must repeatedly run their RSM-based programs on the robot and watch how well it performs. They must analyze the failures and determine how they emanate from the RSM. They must then modify the RSMs in order to correct the failures or improve the performance. During this debugging period, students learn a great deal about the interaction between the robot's sensors and its physical environment, and in turn must translate this knowledge back into their programs and behaviors. The goal of this project is to design machines that will react to sound, temperature, touch, words (text) from speech recognition, simplified image recognition, light sensors, etc. Here, we want to design something like the famous Furby toy, but ***with real learning***.

Let us first discuss how Furby works. It can be observed that Furby's internal states are prespecified, its learning is only transiting to prespecified new states in the labyrinth of its possible "states of emotions and knowledge" (sleeps, plays games, sings, is ill/healthy, etc). Appropriate learning patterns (such as petting the head twice and next the back once) lead to the displays of appropriate behaviors pre-stored in the toy's memory, and are only hidden from the user by not entering some internal states earlier. Although this is not a real learning, it is perceived as astounding by people who observe this toy. Now, we want to create a puppet robot similar in sensor/actuator Pavlovian/Skinnerian learning model, but that will built its "world model" with unlimited behaviors. This model is by inducing the logic of prespecified behavior hierarchy of RSMs. The totally new states with their respective input/output behaviors will be created using our approach. In contrast to Furby, our robot has ***new*** internal states, created automatically, and not known to the designer. The power of constraint-based RSM descriptions induced from examples of our approach will be again used to achieve this task. Instead of only transiting to "higher levels of consciousness" as Furby, the robot will create its own space of internal states and transitions, modeling a simplified environment. Observe however, that there is still no real world involved directly in learning. The observation of the results of robot's behavior is done by the student-supervisor and it is him who in a God-like manner (or, brain-surgeon manner) inserts directly knowledge to the software/silicon brain of the robot (the set of RSMs). In this world, there exist two kinds of learning. The learning in the software, which is designing the robot's brain, and the learning of the student, which means, the student learns to invent good learning sequences. Observing how robot reacts to the sequences, he invents new learning sequences that directly affect robot's brain. The student has the power of directly changing the brain of the robot, as he wishes, and according to what he observes in the real world, but there is no mechanism of having the world tell directly to the robot what is wrong with its actions. This would require the evolution and the evolving robot, which is done in other learning methods.

5. Evolvable Robot that learns from its mistakes in environment

In this approach the robot's mechanics is fixed, and there is now no human supervisor. The real world or the environment, serves now as the supervisor. This is the most complex of the learning modes. Robot executes the genetic algorithm; with its cycle of parents' chromosomes crossover, mutations, fitness functions, and the survival of the fittest. Having, however, only one robot, we will be growing only the brains of robots in the robot population (i.e. each

reactive state machine plays the role of one chromosome) and the populations of these machines will be tested/evaluated in real mechanics/hardware. Thus, a programmed robot's genotype is its hierarchy of reactive state machines. The programmed robot's phenotype is its physical robot with its brain in the real physical environment of robotic hand and head, external items and obstacles, that punishes it or rewards for its actions. At the beginning of its life the robot phenotype obtains certain number of units of energy. With time this energy is dissipated (the robot is getting old), and the robot is supposed to achieve some tasks (like bringing a box from place to place). For each task achieved the robot phenotype (its part of memory other than the RSM) is rewarded (a number is added to the value of the fitness function). For each task not achieved, or a mistake (like bumping an obstacle with a hand) the robot is penalized (a number is subtracted from the fitness function). When certain given time passes (i.e. when the robot dies), the total fitness function of the robot is known. If it is small the robot's genotype is send to eternal damnation. If it is high, the robot is allowed to reproduce, which means, new reactive state machines will be created by reproducing the most successful genotype RSMs. Thus the corresponding child genotype and next child phenotype robot are created. The child will be tested again in the same environment (reincarnation?). Thus the supervisor is removed from the loop now, the human provides only the formulation of the fitness function. It will be up to the student to define the fitness function and to develop experiments. He will evolve the robot as a series of RSM files, created by positive and negative feedbacks from the physical environment. We just started work on this method and so far it has been not programmed to BUG robot but is developed for robot soccer.

6. Simple Example

Given are two sensors, left sensor LS and right sensor RS located in front of a mobile robot with two wheels and a caster wheel. Left wheel is controlled by motor LM and right wheel by motor RM. Both motors can run only forward. All variables are ternary. LS=0 means no light, LS=1 means weak light and LS=2 means strong light perceived by sensor LS. Similarly, for sensor RS. LM = 0 means LM stops, LM =1 is slow movement of LM, LM=2 is fast movement of LM. Similarly, for motor RM. The sensors are connected to inputs of the black box and the motor controls are connected as outputs of the black box that we are going to teach, in this case teaching is inducing a combinational multi-valued function of robot's behavior. Random output signals are created by some additional mechanism to be able to create learning conditions and at the same time certain information from sensors is perceived by the system. Robot executes this some random behaviors such as stopping, turning left quickly, going forward slowly, going forward quickly, etc. which are perceived by the supervisor – a child teaching the robot. Each of the behaviors is evaluated as good or bad using human's voice. All positive reinforcements are used to create an object/attribute table from Figure 2. Only those input/output combinations of LS, RS, LM and RM that got human praise are stored. This table plays then a role of a characteristic function of a relation. As a result of synthesizing the MV functions LM = F1(LS,RS) and RM = F2(LS,RS) using the bi-decomposer, functions F1 = MAX(LS,RS) and F2 = MIN(LS,RS) are created. Observe that many other relations (and particularly functions) can fit the data of the characteristic function from Figure 2, and thus many different rules can be induced from the provided examples and their reinforcements. But the solution above is the simplest solution, thus satisfying the Occam Razor principle of learning. The synthesize rule means that if both sensors give no light, the robot stops. With small lights from both sensors robot moves forward slowly, with strong lights from both sensors robot moves forward quickly. Robot turns always right in case of unequal strengths of light sources in sensors. Turning right is faster and sharper if the difference of light strengths is larger. Many similar examples of light following and light avoiding robots can be induced from relations. The same principles are applied to robot head and hand movements, but much more complex functions are induced since there are many sensors , motors, timings, voice and vision. The method can be generalized to stochastic relations [1].

7. Conclusions

There are ***two aspects*** of the research and development presented here. First, we introduced a *new method of teaching new behaviors to robots.* This method is an adaptation of approaches used in the past for data mining and circuit design. By this, we demonstrated that the previously introduced by us "learning by relation decomposition" is a truly general method of machine learning. We plan to use this method also for other robot learning tasks such as learning walking gaits for a hexapod robot [8]. Second, we proved that advanced robotics can be taught with success to high school students. Please look to figures below and [18] for photographs of our robots and more technical details of this project.

8. Literature

1. A. Al.-Rabadi, M. Zwick, and M. Perkowski, “A Comparison of Enhanced Reconstructability Analysis and Ashenhurst Curtis Decomposition of Boolean Functions”, *12th International WOSC Congress and the 4th International Institute for General Systems Studies Workshop*, Pittsburgh, Pennsylvania, USA, March 24-26, 2002.
2. R.A. Brooks, ``A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation,* 2(1):14, 23,March 1986.
3. M. Domsch, ``MIT 6.270 LEGO Robot Design Competition," *World Wide Web*, URL ishttp://www.mit.edu/courses/6.270/home.html.
4. C. Espinosa, “Low Level Image Processing for Mobile Robots”, M.S. Thesis, PSU.
5. J. Iovine, ``Robots, Androids, and Animatrons. 12 Incredible Projects You Can Build." *Mc Graw Hill,* 1998.
6. C. Files, and M. Perkowski, “New Multivalued Functional Decomposition Algorithms Based on MDDs”, *IEEE Transactions on CAD*, Vol. 19, September 2000, pp. 1081-1086.
7. S. Grygiel, M. Zwick and M. Perkowski, “Multi-level decomposition of probabilistic relations”, *12th International WOSC Congress and the 4th International Institute for General Systems Studies Workshop*, Pittsburgh, Pennsylvania, USA, March 24-26, 2002.
8. M. Levy and M. Perkowski, “Gait generation for a hexapod robot via functional decomposition*”, to be submitted*.
9. A. Mishchenko, C. Files, M. Perkowski, B. Steinbach, and Ch. Dorotska, ``Implicit Algorithms for Multi-Valued Input Support Manipulation," *Proc. of 4th Intl. Workshop on Boolean Problems,* September 2000, Freiberg, Germany, pp. 9 - 20.
10. A. Mishchenko, B. Steinbach, and M. Perkowski, ``An Algorithm for Bi-Decomposition of Logic Functions," *Proc. Design Automation Conference, DAC 2001,* June 18-22, Las Vegas, pp. 103 - 108.
11. A. Mishchenko, B. Steinbach, and M. Perkowski, ``Bi-Decomposition of Multi-Valued Relations," *Proc. 10-th International Workshop on Logic and Synthesis, IWLS'01* , pp. 35 - 40, Granlibakken, CA, June 12 - 15, 2001, IEEE Computer Society and ACM SIGDA
12. H. Moravec, ``Robot. Mere Machine to Transcendent Mind," *Oxford University Press*, 1999.
13. R.E. Murphy, “An Introduction to AI Robotics”,
14. M. Perkowski, S. Grygiel, Q. Chen, and D. Mattson, ``Constructive Induction Machines for Data Mining," *Proc. Conference on Intelligent Electronics, Sendai,* Japan, 14-19 March, 1999. Invited speaker.
15. M.A. Perkowski, A.N. Chebotarev, A.A. Mishchenko, ``Evolvable Hardware or Learning Hardware? Induction of State Machines from Temporal Logic Constraints," *The First NASA/DOD Workshop on Evolvable Hardware (NASA/DOD-EH 99),* Jet Propulsion Laboratory, Pasadena, California, USA, July 19-21, 1999, pp. 129 – 138.
16. M. Perkowski, ``Oregon Cyber Theatre,*'' Proc. 3rd Oregon Symposium on Logic, Design and Learning,* May 2000.
17. M. Perkowski, and S. Grygiel, `` Decomposition of Relations: A New Approach to Constructive Induction in Machine Learning and Data Mining - An Overview" *Proc. Workshop of National Institute of Telecommunications and Polish Association for Logic and Philosophy of Science,* May 25, 2001, pp. 91 - 111, Warsaw, Poland.
18. M. Perkowski’s webpage, http://www.ece.pdx.edu/~mperkows/
19. S. Russell and P. Norvig, ``Artificial Intelligence: A Modern Approach," Prentice Hall, Englewood Cliffs, NJ, 1995.
20. W. Savitch, “Problem Solving in C++,”
21. B. Steinbach, M. Perkowski, and Ch. Lang, ``Bi-Decomposition in Multi-Valued Logic for Data Mining," *Proc. ISMVL'99*, May, 1999, pp. 50 – 58.
22. S.E Umbaugh, “Computer Vision and Image Processing”,
23. www.codeproject.com.
24. www.freshmeat.com. This website offers a variety of source code and activeX controls available for download.
25. The bi-decomposer of relations and other useful software is available at http://www-cad.eecs.berkeley.edu/mvsis/. See also several papers by R. Brayton and his coworkers in IWLS 20

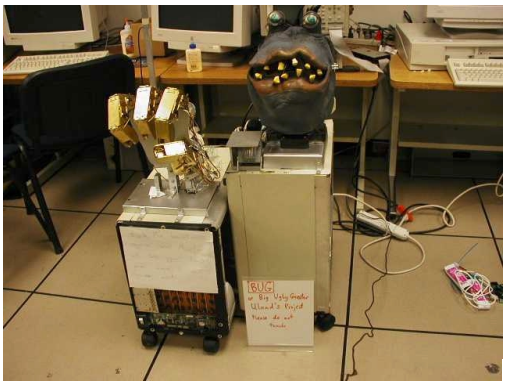

Figure 1. Big Ugly Robot BUG

LS	RS	LM	RM	reinforcement
0	0	0	0	yes
0	1	1	0	yes
2	2	2	2	yes
1	1	1	1	yes
2	0	2	0	yes
1	2	2	1	yes

Figure 2. Teaching a mobile robot to react to light sources

Figure 3. Head can talk and move towards you

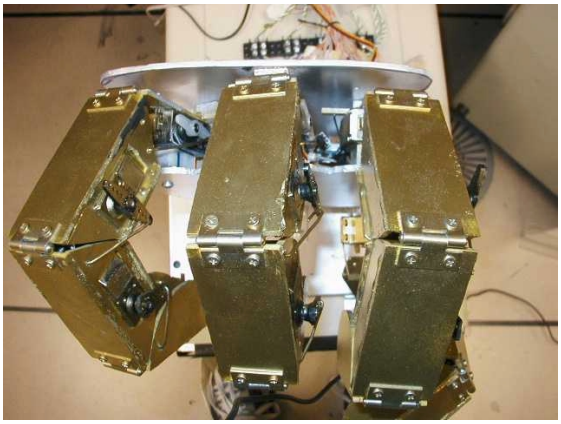

Figure 4. Details of a four-fingered hand

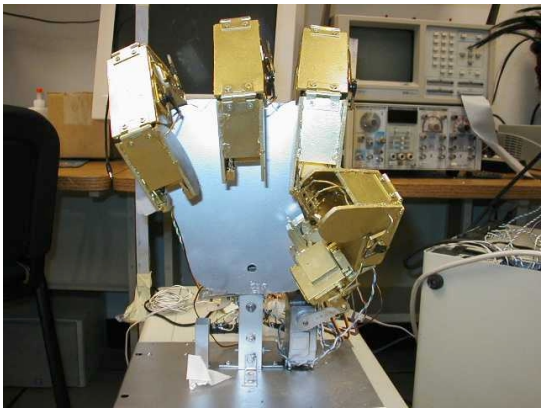

Figure 5. Each finger is controlled by 4 servos. There is a total of 16 servos in the hand. This allows for many programmable gestures

Figure 6. Uland Wong as seen by the robot camera. Black hair is easy to find and locate.

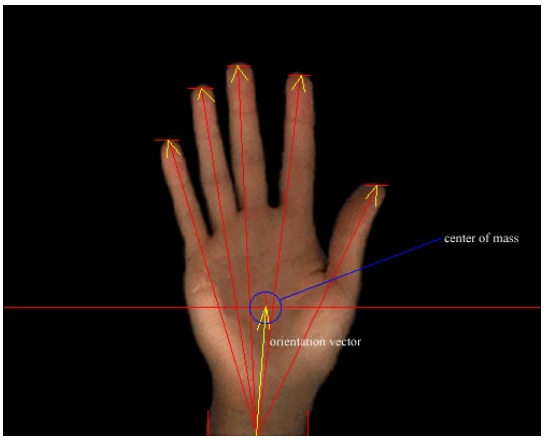

Figure 7. Locating fingers.

Figure 8. Locating hand or face.