FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 20, no. 3, December 2007, 561-586

Inductive Learning of Quantum Behaviors

Martin Lukac and Marek Perkowski

Abstract: In this paper studied are new concepts of robotic behaviors - deterministic and quantum probabilistic. In contrast to classical circuits, the quantum circuit can realize both of these behaviors. When applied to a robot, a quantum circuit controller realizes what we call *quantum robot behaviors*. We use automated methods to synthesize quantum behaviors (circuits) from the examples (examples are cares of the quantum truth table). The don't knows (minterms not given as examples) are then converted not only to deterministic cares as in the classical learning, but also to output values generated with various probabilities. The Occam Razor principle, fundamental to inductive learning, is satisfied in this approach by seeking circuits of reduced complexity. This is illustrated by the synthesis of single output quantum circuits, as we extended the logic synthesis approach to Inductive Machine Learning for the case of learning quantum circuits from behavioral examples.

Keywords: Quantum circuits, machine learning, logic synthesi, quantum robot behavior.

1 The Concept of Learning Quantum Behaviors From Examples.

It is well-known that logic synthesis methods applied to binary functions with many don't cares (don't knows) are used as a base of various machine learning (ML) approaches [1, 2, 3]. The learning process creates circuit description and as a byproduct converts don't cares to cares trying to satisfy the Occam Razor Principle of the circuit's simplicity. While the method of logic synthesis based machine learning was already applied to binary and multiple-valued circuits [1, 3], here it is applied for the first time to quantum circuits [4, 5, 6].

It is well-known that an Einstein-Podolsky-Rosen (EPR) circuit [4, 7] composed of a Hadamard gate and a Feynman gate realizes entanglement. In an ex-

Manuscript received August 18, 2007.

The authors are with Portland State University, Portland, Oregon, USA (e-mails: lukacm @ece.pdx.edu, mperkows@ee.pdx.edu).

tended EPR circuit the Hadamard gate can be additionally controlled, which means that when controlled with signal $|0\rangle$, the EPR circuit changes to a single Feynman gate and the entanglement is removed, thus the circuit's behavior becomes deterministic. Similarly the controlled Hadamard and Controlled Square-Root-of-Not (CV) gates can be used as sources of randomness when the state of the circuit's output is measured [4]. This way we designed several interesting circuits and used them to control the behaviors of various kinds of robots [8].

A robot controller is a mapping between inputs (sensors) and outputs (actuators). So the mapping is closely related to the behavior observed on the robot. If the mapping is specified by a unitary matrix it is a controller of a quantum robot. These behaviors combine quantum-probabilistic and deterministic - we call these *quantum robots* and we say that they exhibit *quantum behaviors*, [8, 9]. From now on we will not distinguish between the quantum-controlled robot, its quantum circuit controller (unitary matrix) and its behavior.

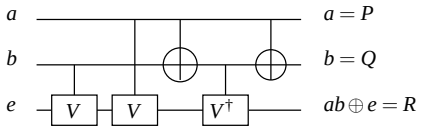


Fig. 1. Toffoli gate realized using 2×2 controlled quantum gates. When used as a quantum robot controller, signals a,b and c can come from touch, sound or other sensors and outputs P, Q and R through measurement units go to motors or other actuators.

The well-known quantum realization of the reversible Toffoli gate (Fig. 1) using Controlled-NOT (CNOT), Controlled-V (CV) and Controlled-$V^\dagger$ ($CV^\dagger$) gates [10, 11, 4] is another source of inspiration because it shows that a deterministic behavior of a permutative quantum (classical reversible) circuit is created using truly quantum gates (such as Controlled-V) that operate in Hilbert Space and with intermediate signals that are superposed [4]. By truly quantum gates we understand those that their unitary matrices are not permutative. If we would thus measure the data path signal in the lowest qubit in Fig. 1 in the middle of this circuit, after two CV gates controlled by inputs a and b respectively, the behavior would be deterministic for some input signals and probabilistic for other ones, leading to very interesting behaviors of a Quantum Braitenberg Vehicle [8] controlled by this circuit. Even more complicated binary quantum circuits (with permutative unitary matrices) can be composed from gates that are the controlled Pauli X rotations by angles π/k where k is a power of two. This leads to gates such as NOT - 180° rotation, square-root-of-not - 90° rotation, fourth-order-root-of-not - 45° rotation, etc. Gates that rotate by $k*(2\pi/3)$ where k is an integer are used in ternary quantum logic. These all rotation gates can be controlled by arbitrary quantum states

[5]. When the resultant signal in the data path bit (the controlled qubit) is an eigenvalue of the unitary transformation(s), the behavior is deterministic. When it is not, the behavior is probabilistic according to the rules of quantum measurement [4, 7]. This means that a system in a superposed state, when measured, collapses to one of the possible observables given by the measurement operator. This way, a circuit can be designed from a set of examples corresponding to the care minterms of a truth table. For instance, value 0 may correspond to sensor conditions when we want our robot to turn left, and value 1 to the true minterm of input variables (a positive example) when the robot should turn right. Based on his design goals the designer specifies examples of robot behaviors as input-output pairs. The software induces behaviors for all other input states that are possible.

With the above background it is now possible to define the principles of inductive learning used in this paper.

Definition 1.1 Inductive Learning in Logic Synthesis for completely Specified functions
Let I be a set of vectors such that $I_k^p = \{i_0, i_1, \ldots, i_n\}$, $n = 2^N$ is the k-th input vector (of N qubit) of pattern P (or function specification) and $f : I \rightarrow O$ be a reversible function, with $O_k^p = \{o_0, o_1, \ldots, o_n\}$ being the expected result vector for the input pattern I_k^p and O is the set of all output vectors.. Let, $i_k \in \{0,1\}$ and $o_k \in \{0,1\}$ be the elements of the input and output vectors respectively and $\sum_{k=0}^{2^N} |\alpha|_k^2 = \sum_{r=0}^{2^N} |\alpha'|_k^2 = 1$ specifying the $l_2 -$ norm space. Let $|\psi\rangle$ be a 3-qubit quantum state and G be the set of possible operators (quantum gates). Then there exists a quantum logic circuit U_f such that for any pair of input and output patterns (I_i^p, O_i^p); $I_i^p \in I^p$, $O_i^p \in O^p$ where $\forall O_i^p \in O \exists I_i^p \in I$ such that $f(I_i^p) = O_i^p$ is a one -to-one mapping. For quantum learning this means that there is a unitary transform on a quantum system $U_f|\psi\rangle \rightarrow |\psi'\rangle$ for $|\psi\rangle \in I^p$, $|\psi'\rangle \in O^p$. The learning of such a function implies to find the minimal set of quantum gates implementing function f (and realizing unitary matrix U_f).

Example 1.0.1 Completely specified reversible function realized in quantum logic
The verification of the above definition is simple because the definition implies a permutative function mapping to which directly corresponds a single unitary transform (which is also permutative). Let f be a completely defined function represented by the Karnaugh Map in Table 1.

Then one of possible realizations of function f is shown in Figure 2. The eight cells of the K-map from Table 1 correspond to eight input-output patterns. Thus input pattern $abc = |110\rangle$ is mapped for instance to the output pattern $PQR = |100\rangle$, etc. As we see this function is reversible as it is a one-to-one mapping (set of output

Table 1. A K-map of a completely specified 3×3 reversible function

c / ab	0	1
00	000	001
01	011	010
11	100	101
10	110	111

(PQR)

patterns is a permutation of the set of input patterns). Using matrix multiplication and Kronecker products of the elementary matrices of all 2×2 gates involved, one can verify that mappings of all cells (shown by the permutation of the inputs in Table 1) are satisfied [4, 12]. The circuit in Figure 2 is thus the result of learning (synthesis) from the initial set of examples (Table 1). In this case there are many circuits to satisfy all input-output pairs, but they all have the same unitary matrix.

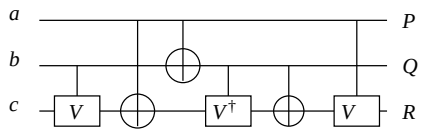


Fig. 2. Example of a completely specified 3×3 reversible function realized as a quantum circuit using quantum primitives Controlled-V (CV), Controlled-V† CV† and Controlled-NOT (CNOT).

Although the Example 1.0.1 was given for completeness and to show the link between logic synthesis and learning, very rarely in real life the system learns (generalizes) from a complete specification(an exhaustive set of examples). In case of robotics it is so only for very small number of sensors and their states.

Definition 1.2 Inductive Learning in Logic Synthesis for incompletely Specified functions
Let I be a set of input vectors defined as in def. 1.1 and let O be the set of output vectors such as in def. 1.1 but with $o_k \in [0,1,-]$. The symbol '-' represents a don't care and corresponds to an unknown output. The set of examples is given as a set of pairs $P = \{i_k, o_k\}$, $k = 1, \ldots, n \leq 2^N$. The inductive learning for incompletely specified functions can be defined as: the process of explicitly finding such a mapping or function satisfying each pair (I_k^p, O_k^p) from the given set P such that $f(I_k^p) = O_k^p$. The result of learning is thus a circuit that describes a complete mapping that agrees with the set of input-output pars from the specification examples.

Example 1.0.2
Let f be a 3-qubit incompletely specified reversible function defined by the Table 2

(it can be checked that the function can be completed to a reversible map since all output care cells $\{000, 001, 100, 101\}$ are different). Table 2 represents thus the set P of learning examples called also the problem specification.

Table 2. An incompletely specified reversible function f

c / *ab*	0	1
00	000	001
01	-	-
11	100	101
10	-	-

Then an arbitrary unitary transformation U that satisfies all the specified transitions,

$$\begin{aligned} U|000\rangle &\rightarrow |000\rangle \\ U|001\rangle &\rightarrow |001\rangle \\ U|110\rangle &\rightarrow |100\rangle \\ U|111\rangle &\rightarrow |101\rangle \end{aligned} \tag{1}$$

together with its corresponding circuit is a valid solution to the learning problem specified in Table 2. Thus the circuit from Figure 2 is a solution also to the learning problem specified in Table 2. Let us observe that in this case there are not only many circuits that solve this problem but also many unitary matrices. Because of Occam Razor the circuit is reduced and as a byproduct its unitary matrix is simplified as well.

To complete the definition of the Learning of an incompletely specified function let us have a closer look at the don't cares.

Lemma 1
Any quantum-permutative function being build according to the above Inductive Learning method and for arbitrary quantum basis state $|I\rangle$ in complex Hilbert space $H^{\otimes^N}$ from a set of single-qubit and two-qubit qubit operators (such as for example G = {[*I*], [Controlled-V], [Controlled-V†], [Controlled-NOT]}) will result in a completely specified function allocating the unknown elements according to the unitary evolution matrix defined such as

$$U|I\rangle = |O\rangle, \tag{2}$$

where $|O\rangle$ is the binary basis output state vector. For every other state $I'\rangle$ there exists a unique quantum state $U|I'\rangle$.

To prove Lemma 1 is is sufficient to observe that we select such gates that the unitary matrix of their compositions (using Kronecker product for parallel connections of gates and matrix product in reverse order for serial connections of gates) is a standard unitary matrix (with no don't cares). This matrix is created in such way that for every vector $|I_i\rangle$ from the pair (I_i, O_i) we have that $U|I_i\rangle = O_i$. Applying matrix U to an arbitrary other input vector $|I'\rangle$, superposed or basis, produces certain output vector $|O'\rangle$ space (in general of complex numbers) so that $U|I'\rangle = |O'\rangle$. This vector $|O'\rangle$ is a completely specified quantum state in a sense that it is a quantum state that is known and deterministic (expressed by a wave equation). On the other hand, after the measurement, there may be very many classical states to which this state $|O'\rangle$ will collapse.

We see thus here a difference between classical and quantum learning. In classical learning we learn a deterministic function. In quantum learning we learn a quantum unitary mapping to a quantum state that is quantum-deterministic only before a measurement, but the observer never knows to which classical state this deterministic state will collapse as the result of the measurement. The designer of the robot sets thus certain constraints for robot's behavior but he can only probabilistically predict how the robot will behave within these constraints.

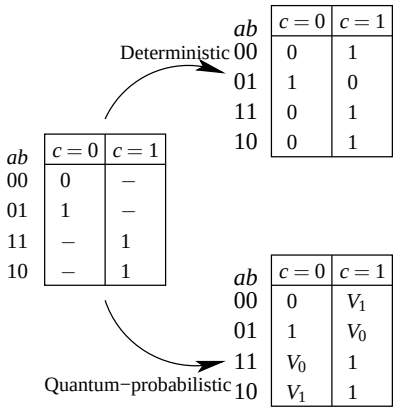


Fig. 3. Two modes of learning based on the properties of quantum systems. On the left the desired single-output function is represented using cares and don't cares. The top Karnaugh Map represents the result of the mapping being the deterministic learning. The bottom K-map represents the learned function using the quantum probabilistic learning.

Taking into account the above introduced quantum phenomena, quantum logic design and the inductive learning, the general mapping of the don't cares to cares in inductive learning can be separated into two categories. Figure 3 illustrates the two learning methods for a single output Boolean function. The Boolean function to be

learned is represented on the leftmost Karnaugh map (K-map) with cares (desired output values from examples) and don't cares (that correspond to cases not known). To make this function reversible the input qubits are forwarded to outputs and a single ancilla bit is added (which is typical for quantum oracles [13]. The first type of learning is shown on the top of Figure 3 and it will be referred to as the Deterministic Learning (it corresponds to the classical learning of Boolean functions). The output of the deterministic learning process is a completely defined Boolean function or a complete reversible Boolean function, if required (Example 1.0.2). The second type of learning called the Quantum-Probabilistic learning, is shown on the bottom of the Figure 3 and has similar results as standard probabilistic learning, with the difference that the probabilities are calculated from quantum states (complex vectors) V_0 and V_1 (the values V_0 and V_1 will be explained later) according to the measurement operation. This paper is focused on explaining the second type of learning; the quantum probabilistic behavior learning for single output Boolean reversible and quantum functions. This is because of its high interest in robotics where we need to specify symbolically deterministic quantum states with expected probabilities of their measured binary outcomes.

Quantum robot has been introduced by Benioff [14, 15, 16, 17] and is described as a quantum system exploiting the superposition and the entanglement of the state of the robot with the state of its environment. Recently in [18] presented are quantum robots with respect to the quality and speed of decision making using the Grover quantum search algorithm [13]. Unlike in these works, here the focus is not on how the robot is implemented with respect to its environment, but rather on the strategies for learning the robotic behaviors based on quantum circuit structures. From the Machine Learning aspects, quantum computing has been already studied in [19, 20, 21, 22, 23] and shows expected speedup of quantum learning with respect to its classical counterpart problems. The machine learning introduced in this paper is focusing on how the structure of the quantum system can be used to build quantum circuits usable as behavioral controllers for socially interactive robots.

With respect to emotional social robotics, our previous research introduced the emotional state machine [9], that is being investigated as a controller of a robot. To explain, Figure 4 recalls the basic concept of the Emotional Quantum Finite State Machine (eQFSM). Such machine can be described as mixed quantum-classical machine with both classical and quantum state transition functions. In Figure 4 the classical logic is labeled F and the quantum-unitary is U. The global state of the eQFSM is unitary in order to be able to represent quantum operations carried by the logic block U. The classical state of the machine is a state of smaller dimension than the global state $|\phi\rangle$ and is the only part of the machine that interacts with the environment using the measurement. The quantum behaviors in this particular

model represent only the quantum logic control that is not directly observable in the environment or controllable from the environment.

In paper [9] behaviors with respect to social context and emotional expression of a social robot built from eQFSM's were analyzed and here only the methods of inductive learning will be explained. In particular the interactions on the quantum level are shown

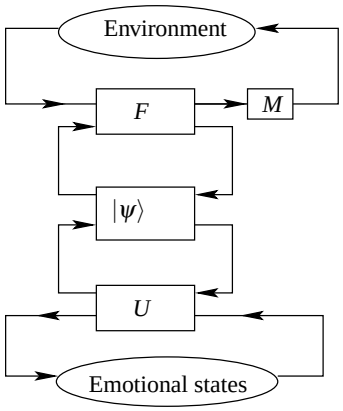


Fig. 4. Emotional Quantum State Machine. The emotional component represented by the logic block U is connected in parallel with other robots emotional components. The logical part represented by block F, interacts either with the environment or with other functional elements. This logical block also performs the measurement on the output states (block M). From the point of view of realization technology both F and U are quantum, but F is described as a permutative unitary matrix, like an oracle in Grover Algorithm, while U is an arbitrary unitary matrix.

with respect to incompletely specified functions that represent input-output behavioral patterns. As already mentioned, the learning process is implemented as inducing the combinational quantum circuit which specifies the whole function and thus maps the 'don't cares' to output values. This circuit is next inserted as U into the eQFSM. In particular, circuits with up to four qubits were analyzed by us in terms of learning. In short, we can say that the classical logic (rational) behavior F of the robot is controlled by emotional behavior U thus being responsible for probabilistic behaviors of the robot. With an appropriate learning approach these behaviors should be still somehow rational and explainable. For instance, a "hysteric" robot will bump more often into obstacles than a "cool" robot would, but it should still follow the light, if so specified by the expected logic behavior. It is similar to a behavior of a person under influence of emotional stress that behaves differently in an observable way but still within certain logic of the situation.

The robotic behavioral framework also requires a certain adaptation of the robot

with respect to the environment. Some of its actions might be required not to be performed exactly as ordered or even that the robot should inductively learn something from its environment. Thus when learning robotic behaviors it is possible to allow a certain percentage of wrong outputs. As will be seen below in the quantum behavior-learning problem it is acceptable that some small percent of cares in the solution is not in agreement with the symbolic cares of the initial specification.

The inductive learning process presented here learns the entire description in one run (in contrast to incremental learning). Thus machine learning is the same as logic synthesis. The synthesis process preserves the cares but replaces all don't cares (don't knows) with deterministic, probabilistic or entangled states. The measured probabilities of outputs result from the circuit's structure and the types of controlled 2×2 gates used by the synthesis algorithm (here, we use gates that are controlled roots of unity with various angles).

Definition 1.3 Quantum Logic synthesis
The synthesis problem is to find the simplest circuit for a truth table with (usually few) given cares (examples) and (usually many) don't cares. Let G be a set of single-qubit and two-qubit unitary operators on complex H^N, then the process of synthesis can be expressed as a minimization of the given function with respect to the width of the circuit and the amount of elementary operators used. Thus it can be written

$$S_{H^N}(n,G) \xrightarrow{min} V(n,G) \tag{3}$$

where $V(n,G)$ is the cost of the circuit constructed of gates from set G.

The minimization of the cost of the circuit with respect to the logic function can be studied using a genetic algorithm (or exhaustive search for small number of variables). The automatically synthesized controllers (i.e. learned from examples) produce very interesting and often unexpected but correct robot behaviors [8, 9]. Moreover, the method presented here uses truly quantum gates; unlike in other papers on quantum logic synthesis here only single-qubit and two-qubit gates are used. Thus, the circuits learned by this method are directly implementable in quantum hardware and they satisfy Occam Razor with respect to the real hardware costs. Then, we can say that these behaviors are more natural to quantum world than behaviors learned with many-input Toffoli gates that would push the solutions towards less probabilistic behaviors or to no probabilistic behaviors at all. This is again more similar to human learning that includes always some probabilistic component (at least as related to body motions and to speech).

2 Symbolic Quantum Synthesis

Assume a single output function defined by a Karnaugh map specifying the desired outputs as would result by observing values 0 and 1 on the quantum output in some special cases of the state of the environment as shown in Table 3.

Table 3. Single output Boolean functions R to be synthesized (learned)

c / ab	0	1
00	-	-
01	0	-
11	1	0
10	-	1

R

c / ab	0	1
00	-	-
01	V_0	V_1
11	-	-
10	-	-

R

As already specified, the quantum circuits may generate outputs with certain probability p. The Table 3a has half don't cares and half cares, and assuming there is a method to synthesize the cares, the problem that remains to be solved is in what manner it is possible to specify the values of don't cares. The unitary operators used in this work are mainly [W] (Wire/Identity), [NOT], [V], [V†], [CNOT], [CV] and [CV†] and thus it should be specified how the don't cares are filled with respect to inductive learning. For this, let $S_o = \{0,1,V_0,V_1\}$ be the set of all possible (symbolic) outputs of the given single output function (qubit R in Fig 5). Then, 0 and 1 represent 100% probability of obtaining 0 and 1 respectively after observing (i.e. measuring) the system's output. $V_0 = V|0\rangle$ and $V_1 = V|1\rangle$ are symbols that represent quantum states (vectors of complex numbers) that corresponds to measuring/observing the system in state $M(V_0)$ and $M(V_1)$ with M being the quantum projective-measurement operator. The symbol '-' is a don't know or can also be seen as a non-observation/measurement of the system and exists only in the initial problem specification. An example of an obtained result using automated synthesis is shown in Figure 5 and as will be discussed later, its behavior is deterministic for the desired qubit R, while the global state might be in a superposition of states. Let us note that when all controls of gates are binary the possible quantum states are only 0,1, V_0 and V_1 (this was proven in [11]).

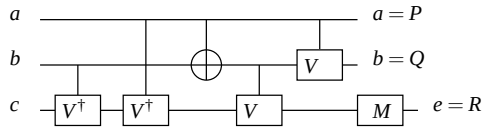


Fig. 5. Example of a solution to the learning problem for the incomplete function from Table 3. Although the circuit as a whole is not deterministic, its behavior observed on qubit R is deterministic.

It is possible to synthesize any single-output Boolean function using the above set of gates with the expected value of the outcome specified by S_o. Proof for this has been already shown in other works [24, 11] however the important point is the methodology. A systematic application of the above gates to the correct qubits will realize this function. Proofs of this methodology for quantum-permutative multi-output functions were given in [24, 11] where the SAT-based search method was presented for exact minimum solutions. In the new paper the above function was realized using a Genetic Algorithm presented in [25, 26], and the learned the circuit from Fig. 5 with output function for a single output variable R takes the following form $R(a,b,c) = [0,1,0,1,0,1,1,0]$ replacing all don't cares of the initial specification (Table 3a) with either 0 or 1. In this case, although we did not restrict the synthesis to deterministic circuits the deterministic behavior was found.

One could however desire to restrict the specification of a gate or a circuit by quantum related symbolic values. Using the above set of gates and controlling the gates only with binary signals it can be seen that all output symbols from the initial specifications should be converted to quantum cares as shown in Table 4. This means that if one would replace the Controlled-V operators by Controlled-Hadamard, the expected input-output mapping will be changed accordingly, despite the fact that after measurement the output values might be the same. For instance, for an input combination 010 (Table 3b) the user specifies quantum state V_0 which after the measurement will produce 0 or 1 each with $\frac{1}{2}$ probability of being observed. For input 011 the symbolic state is V_1. This would produce the same output after direct measurement as state V_0, but has a different phase, which can be used if the value of the output qubit is used in some other circuit.

Table 4. Possible mapping between input specifications and their symbolic representation as a result of learning (learned circuit); each well defined input value (0, 1) is to remain as it is. The don't care input specifications can be mapped up to four different symbols: 0, 1, V_0, V_1. The "-"s will be thus replaced with 0,1, V_0 or V_1.

0	$\rightarrow$	0
1	$\rightarrow$	1
V_0	$\rightarrow$	V_0
V_1	$\rightarrow$	V_1
-	$\rightarrow$	$0,1,V_0,V_1$

For example the incompletely specified function (defined in Table 3a) could be learned to the completely quantum-defined function $f = [V_0,V_1,0,V_1,V_0,1,1,0]$ as a result of the synthesis process and as such would still preserve the requirements as well as its probabilistic behavior for certain inputs.

Thus the probabilities of the output states directly depend on the logic elements (quantum gates) selected by the synthesis algorithm. If for example we would extend the set of gates to include also the gates $[\sqrt{v}]$, $[\sqrt{v^\dagger}]$, $[\sqrt[4]{v}]$ and $[\sqrt[4]{v^\dagger}]$, the

output state values would be appropriately changed and other probabilities than $\frac{1}{2}$ would be possible.

It is even more interesting that with respect to the specification from Table 3a the circuit from Figure 5 has a single deterministic output under measurement (as required), but its global three-qubit state is in superposition. For example for state $|110\rangle$ transition is: $|110\rangle \xrightarrow{U} \frac{1-i}{2}|101\rangle + \frac{1+i}{2}|111\rangle$, but the first and last qubits are unaffected by the measurement. Now assume deterministic learning in which the circuit from Figure 6 was found. Observe that it also satisfies the incomplete function from Table 3a for single qubit R.

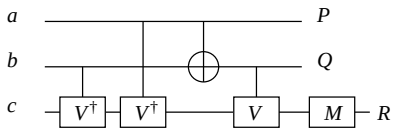


Fig. 6. Example of a deterministic quantum circuit being the solution to the learning problem with the set of examples specified by the incomplete function specification from Table 3a

A more detailed view on the implementation from Figure 6 allows to make a transparent analysis based on symbols $V, V^\dagger$ and their algebra ($VV = NOT$, $V^\dagger V^\dagger = NOT$, $VV^\dagger = V^\dagger V = I$). This is done in Table 5.

Table 5. Example of analysis of the signal R in the circuit from Figure 6

ab c	0	1
00	I	I
01	$VV^\dagger$	$VV^\dagger$
11	$V^\dagger V^\dagger$	$V^\dagger V^\dagger$
10	$V^\dagger V$	$V^\dagger V$

→

ab c	0	1
00	I	I
01	I	I
11	NOT	NOT
10	I	I

→

ab c	0	1
00	0	1
01	0	1
11	1	0
10	0	1

For functions with multiple outputs, the above methodology applies as well but the analysis is more complicated. The circuit from Figure 5 does generate deterministic output for the single measured qubit however has probabilistic behavior from a multiple qubit measurement. For example the transition for the input state $|100\rangle$ can be analyzed as follows:

$|100\rangle \xrightarrow{U}$

apply the Leftmost gates W (wire) and Controlled-V†

$\xrightarrow{W\otimes CV^{\dagger}} |100\rangle$

apply the gates Controlled-V† spanning the

the whole width of the circuit (jumping over the middle qubit)

$\xrightarrow{CV^{\dagger}} |10\rangle\frac{1-i}{\sqrt{2}}(|0\rangle+|1\rangle)$

apply the gates Controlled-NOT and W

$\xrightarrow{CNOT\otimes W} |11\rangle\frac{1-i}{\sqrt{2}}(|0\rangle+|1\rangle)$ (4)

apply the gates W,Controlled-V

$\xrightarrow{W\otimes CV} |110\rangle$

apply the gates Controlled-V,W

$\xrightarrow{CV\otimes W} |1\rangle\frac{1+i}{\sqrt{2}}(|0\rangle+|1\rangle)|0\rangle$

$\rightarrow |1\rangle\frac{1+i}{\sqrt{2}}(|0\rangle+|1\rangle)|0\rangle$

$\rightarrow \frac{1+i}{\sqrt{2}}(|100\rangle+|110\rangle)$

Thus the original three-to-one incompletely specified function from Table 3a is mapped to the reversible quantum three-by-three circuit from Figure 5 to which corresponds the fully specified and deterministic (but probabilistic after measurement) quantum function that is represented in the Karnaugh map from Table 6.

The Table 6 can be formally minimized (allowing the Toffoli function) using the following conditional notation that allows to express the logical and causal dependence between the control and data qubits. For example, to denote a CNOT gate one could write $CNOT \rightarrow (|1\rangle_a NOT)|b\rangle$. To explain, the CNOT gate described shows two qubits: the $|b\rangle$ is the output qubit and the control qubit activates the unitary transform on $|b\rangle$ only when equal to 1. Thus $(|1\rangle_a NOT)$ means the when the control bit is is in state $|1\rangle$ the NOT operation is applied to target qubit. Thus

Table 6. Function mapped by the circuit from Figure 5. For each qubit in the register the unitary evolution is sub-scripted by a, b and c for each qubit respectively.

ab c	0	1
00	I_a, I_b, I_c	I_a, I_b, I_c
01	$I_a, I_b, (VV^\dagger)_c$	$I_a, I_b, (VV^\dagger)_c$
11	$I_a, V_b, (V^\dagger V^\dagger)_c$	$I_a, V_b, (V^\dagger V^\dagger)_c$
10	$I_a, V_b, (V^\dagger V^\dagger)_c$	$I_a, V_b, (V^\dagger V^\dagger)_c$

$\rightarrow$

ab c	0	1
00	I_a, I_b, I_c	I_a, I_b, I_c
01	I_a, I_b, I_c	I_a, I_b, I_c
11	I_a, V_b, NOT_c	I_a, V_b, NOT_c
10	I_a, V_b, I_c	I_a, V_b, I_c

$\rightarrow$

ab c	0	1
00	$\|000\rangle$	$\|001\rangle$
01	$\|010\rangle$	$\|011\rangle$
11	$\|1\rangle \frac{1+i}{\sqrt{2}}(\|0\rangle + \|1\rangle)_b \|1\rangle_c$	$\|1\rangle_a \frac{1+i}{\sqrt{2}}(\|0\rangle + \|1\rangle)_b \|0\rangle_c$
10	$\|1\rangle \frac{1+i}{\sqrt{2}}(\|0\rangle + \|1\rangle)_b \|0\rangle_c$	$\|1\rangle_a \frac{1+i}{\sqrt{2}}(\|0\rangle + \|1\rangle)_b \|1\rangle_c$

the above function can be rewritten to:

$$\begin{aligned} |a\rangle &\xrightarrow{U} I|a\rangle, \quad -Identity \\ |b\rangle &\xrightarrow{U} (|1\rangle_a (V \times NOT))|b\rangle \\ |c\rangle &\xrightarrow{U} (|11\rangle_{ab} NOT)|c\rangle \end{aligned} \tag{5}$$

and the minimized circuit is shown in Figure 7.

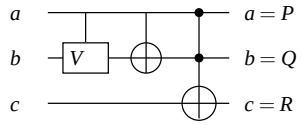


Fig. 7. The final circuit minimized by using the symbolic reduction method to function specification from Table 3

3 Learning Quantum Behaviors

In this section we illustrate few examples of formulating quantum behavior learning problems as quantum circuit synthesis problems according to the Inductive Learning from Section 2. The synthesis of quantum circuits is shown applying a composition method with a restricted set of gates used. This approach is based

on the structured exhaustive and GA-based generator from Fig. 8. Functions f_i are arbitrary Boolean functions and functions g_i are arbitrary quantum rotations. It can be proven that each gate composed of a control function f_i and a data path(target) gate g_i below is a quantum operator specified by a unitary matrix. It is a reversible(permutative) gate when g_i is a binary logic operator (i.e. NOT operator). In the exhaustive approach all control functions f_i from certain class are systematically investigated as well as all data functions g_i from some other class.

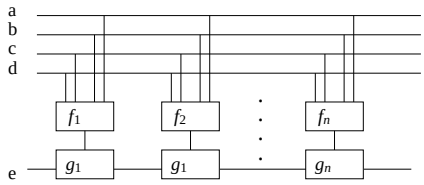


Fig. 8. The general concept of controlled rotation gates for the exhaustive and genetic search-algorithms.

The gates from this class are substituted in all possible ways for f_i and g_i in Fig. 8 and the quantum state is symbolically simulated. The number n of segments is the synthesis parameter and we start from small values to satisfy the Occam Razor. In the GA approach vectors of these functions f_i and g_i are chromosomes. Each chromosome represents a circuit and is built from unitary gates that are available to the GA. The GA minimizes the circuits with respect to the number of gates in each chromosome as well as with respect to the correctness of the implemented circuit.

Example 3.0.3
Let us first look at the well-known Peres gate circuit from Fig. 9.

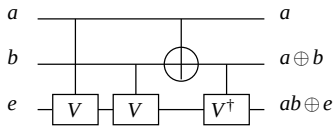


Fig. 9. The Peres gate discovered by the exhaustive and GA algorithms. Its cost is four 2-qubit gates

Many similar circuits were generated automatically using various software approaches in papers [11, 25] but none of them was yet specifically studied and used in behavioral robotics. They use only 1-qubit gates - inverters and 2-qubit gates - controlled-V, Controlled-$V^\dagger$ and Controlled-NOT. Observing these circuits one can appreciate that all controls are linear or affine Boolean functions since only two-qubit permutative gates are allowed (in contrast to the general schema from Figure 8). Thus in our first variant of the general schema for quantum circuits generation all controls f_i were assumed to be affine functions. In case of binary logic, affine functions are linear functions and their negations. Linear functions are constant 0 and XOR's of subsets of the set of input variables. We do not care at this point how the upper part of the circuit, the control f_i, is realized as a minimal reversible circuit - we have developed elsewhere efficient methods for synthesis of such affine functions without ancilla bits. To be precise, single output function F_i is a Boolean irreversible function, in general, arbitrary. It is made reversible by taking into account inputs/outputs a, b, c, d, etc. The controlled functions in the lowest (target)

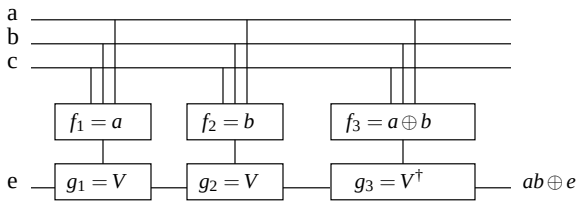


Fig. 10. Applying general schema from Figure 8 to discover the classical Peres structure from Figure 9. Correct substitution values for f_i and g_i (found by the Genetic Algorithm) are shown in boxes. In the solution the input wire c is not used. Note also that the function appears in an output qubit initialized to e.

qubit are inverters, V and $V^\dagger$ gates. This way the 3-qubit Peres gate from Fig. 9 was found from its complete truth table. Figure 10 illustrates how Peres gate can be obtained by applying the scheme from Figure 8 restricted to the length of 3 segments. This was an example of deterministic behavior synthesis from a complete specification of behavior, i.e. (a truth table with all care values). Using this method with limited number of segments some useful new gates have been invented that will be presented in our next papers. Again, Occam Razor leads to the invention of useful new concepts, powerful and inexpensive quantum gates in this case.

Example 3.0.4
Given is a set of examples in the form of a standard incomplete truth table represented by the standard K-map from Table 7. The learning problem is defined as

Table 7. A set of positive (1) and negative (0) examples in a form of standard Karnaugh map of a single output Boolean function.

cd / *ab*	00	01	11	10
00	-	-	1	0
01	0	1	0	1
11	0	0	-	-
10	0	1	0	1

learning the function, i.e. to design a deterministic circuit using only gates CV, $CV^\dagger$ and CNOT and not more than 4 segments. The genetic algorithm was used for the deterministic learning. It found the circuit from Fig. 11. Observe that the symmetric Boolean function $S^{2,3}(a,b,c)$was found as a component of the learned complete Boolean function $f = S^{2,3}(a,b,c) \oplus d = ab \oplus ac \oplus bc \oplus d$. Such symmetric functions result in this and similar problems from the Occam Razor principle. Interestingly, this discovery leads to the same class of functions as those invented by mathematical definitions in an unrelated research by Sasao [27].

The unitary matrix of this circuit is also a permutative matrix and thus its behavior is deterministic. The graphical method of analysis of the solution circuit

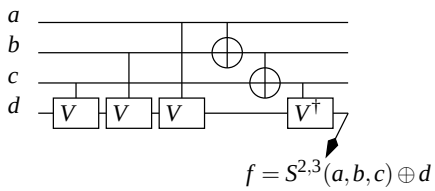


Fig. 11. The final circuit for the majority-controlled gate specified in Table 7

that is applied in the cost function evaluation is shown in Table 8. Identities $V^{\dagger}V = VV^{\dagger} = I$ and $VV = NOT$ were used. The numerical equivalent of this method is used in our exhaustive and evolutionary software as a part of the fitness function calculation.

Table 8. Stages of the symbolic analysis of the Majority Gate Function.

cd / *ab*	00	01	11	10
00	-	-	$VV^{\dagger}$	$VV^{\dagger}$
01	$VV^{\dagger}$	$VV^{\dagger}$	VV	VV
11	VV	VV	$VVVV^{\dagger}$	$VVVV^{\dagger}$
10	$VV^{\dagger}$	$VV^{\dagger}$	VV	VV

↓

cd / *ab*	00	01	11	10
00	I	I	I	I
01	I	I	NOT	NOT
11	NOT	NOT	$NOT \times I$	$NOT \times I$
10	I	I	NOT	NOT

↓

cd / *ab*	00	01	11	10
00	0	1	1	0
01	0	1	0	1
11	1	0	0	1
10	0	1	0	1

Table 9. Example of K-map with half of the values as don't cares

c / *ab*	**0**	**1**
00	0	1
01	-	-
11	1	0
10	-	-

Example 3.0.5
Given is a set of examples represented as cares from K-map in Table 9. All other minterms are don't cares. Use the probabilistic learning method presented in this paper to design a circuit that may have both probabilistic and deterministic behav-

iors. The circuit with three segments was found, Fig. 12a, it has a deterministic output with respect to the desired function: $f = \{000 \rightarrow 0, 001 \rightarrow 1, 110 \rightarrow 1, 111 \rightarrow 0\}$. This mapping agrees thus on all cares with the initial set of examples. The circuit is deterministic and obviously not minimal since the minimal circuit is just $a \oplus c$. In another learning attempt the circuit from Fig. 12b was found with two segments. In this circuit the probabilistic behaviors exist, as can be easily seen. For example $|010\rangle \rightarrow \frac{1}{2}(|010\rangle + |011\rangle)\}$. Analysis of the control for the circuit in Figure 12a using symbolic values is shown in Table 10. Analysis of the circuit from Figure 12b is done in Table 11.

Table 10. K-map for analysis of the learned specification from Table 9 that has a learned circuit from Figure 12

c / *ab*	**0**	**1**
00	$I(0)$	$VV(0)$
01	$I(0)$	$VV(0)$
11	$VV(0)$	$I(0)$
10	$VV(0)$	$I(0)$

→

c / *ab*	**0**	**1**
00	0	$NOT(0)$
01	0	$NOT(0)$
11	$NOT(0)$	1
10	$NOT(0)$	1

→

c / *ab*	**0**	**1**
00	0	1
01	0	1
11	1	0
10	1	0

As it can be seen the K-map from the Table 11 agrees on all cares with the specification from Table 9 but the don't cares from the Table 9 are now replaced with values $V_0 = V(0)$ and $V_1 = V(1)$ which leads to measuring the values 0 and 1 with equal probability $\frac{1}{2}$.

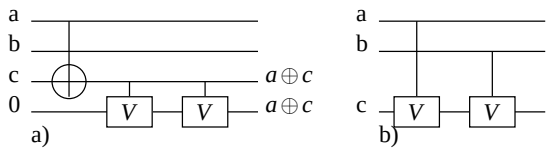


Fig. 12. The final circuits for the function from Table 9. Observe that the circuit 12a is not optimal as the two $Controlled - V$ gates can be combined to a CNOT.

If one would request that the circuit should have a non-deterministic behavior for minterm $\neg abc$ then there would be no choice between the circuit from Fig. 12a and one from Fig. 12b, and only the circuit from Fig. 12b would be a solution to this quantum learning problem.

Table 11. Quantum K-map for analysis of the specification function from Table 9 that has a learned circuit from Figure 12b

c / *ab*	**0**	**1**
00	-	-
01	V	V
11	VV	VV
10	V	V

→

c / *ab*	**0**	**1**
00	$I(0)$	$I(1)$
01	$V(0)$	$V(1)$
11	$NOT(0)$	$NOT(1)$
10	$V(0)$	$V(1)$

→

c / *ab*	**0**	**1**
00	0	1
01	V_0	V_1
11	1	0
10	V_0	V_1

Table 12. K-map of a more complicated function

c / *ab*	**0**	**1**
00	0	0
01	–	–
11	V_0	–
10	0	1

Example 3.0.6
Given is a quantum truth table in a form of a *quantum Karnaugh map* (QKM) as shown in Table 12. Observe that the output values in a QKM can be cares, don't cares and also quantum states such as $V_0 = [V]|0\rangle = V(0)$ and $V_1 = [V]|1\rangle = V(1)$. Design a circuit that has both probabilistic and deterministic behaviors specified by the cares of this truth table. First solution is deterministic, Fig. 13a which leads to K-map from Fig. 13b and the circuit with four segments f_i/g_i(but five gates). Observe that for minterm $ab\neg c$ the value was changed from V_0 to 0 and the symbolic quantum care V_0 is not satisfied. This means that for input combination 110 the robot will always generate 0 and not only in half of the cases. Observe that this circuit does generate the function with a relaxed structure. This means that in the original idea of Symbolic Synthesis the [Controlled-V] and [*CNOT*] gate can only be controlled in the top-down manner; the control qubit is always above the controlled bit. In the circuit from Figure 13 the Genetic Algorithm had relaxed structural constraints on the synthesis and thus the result is a slight deviation form the original paradigm. The second solution circuit, Fig. 14a, leads to the learned

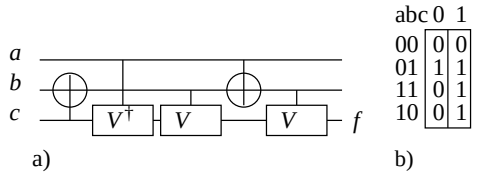


Fig. 13. Solution to non-completely specified quantum function from Table 12. Observe the different structure of the circuit as the requirements are relaxed.

function from Fig. 14b. The circuit has with four controlled-V gate segments and non-deterministic behaviors. In this case all symbolic quantum cares (i.e. V_0, 0 and 1) are satisfied, as can be verified by comparing Table 12 with Figure 14b. Stages of symbolic analysis are shown in Fig. 14c.

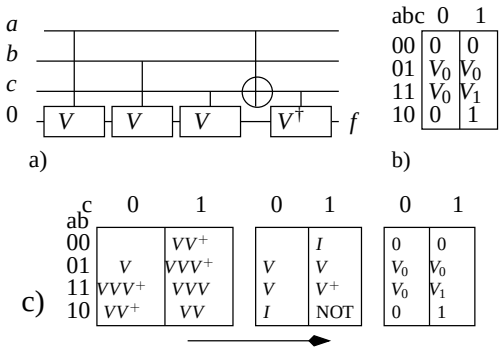


Fig. 14. A more complicated solution to the function from Table 12. a) the circuit, b) the Karnaugh map and c) the analysis of the Controlled-V based logic executed in fitness function calculation. Let us observe that an ancilla bit initialized to 0 was added.

4 Experiments and Results

The genetic algorithm was tested on various benchmarks and for different initial parameter settings. Table 13 refers to various settings and parameters of the various experimentations with the GA. The genetic algorithm was run under all these conditions, and each time a solution was found. From the experimental point of view, there are various parameters that allow to force the GA to particular areas of the problem space. For example, the GA starts with the minimal number of segments equal 2 and the maximal number of segments set up to 5. In another case, the GA will start with the minimum of 9 and the maximum of 15 segments per chromosome. This limitation has been automated to allow the GA explore particular subspace in space with a limited size. In other words, the GA searches for a given solution in a non-limited space, and when a solution is found it tries to find a similar solution in the more size-restricted space. For a given problem that has a known solution with 8 segments, the GA is given freedom to explore surrounding problem space with four segments more than the current known minimum. For more exploration it is possible to adjust the segment number to higher values, so as to restrict it to a more tight search space. Also, when the individual chromosomes are evaluated, if they are too long, they are not to be shortened at the strict requirements given by the user, but the maximum operating length of individual chromosomes is double of the user required. Experimentally this measure was found useful, when searching for a solution to an unknown circuit that the user does not know the minimum representation of. In the case when the maximum circuit length under-estimates the realizable minimum, the solution can still be found due to this double size measure. This can also be observed during the synthesis process. When looking for a circuit

Table 13. GA settings

Population size:	10, 50, 100
Mutation:	0.01, 0.1, 0.3
Crossover:	0.6, 0.7
Fitness function:	Type 3 and 4
Input gate set:	CNOT, CV(V^*), Full
Min/Max Segment size:	(2/5) 3/6, 6/12, 9/15

of a known length n (and assuming this length is minimal) it is required to allow the GA to search space above and below the given length. From previous work in this area [28, 12] it was shown that a "messy GA" algorithm (with random length of the individual circuit (chromosome)) was not completely successful because of the many local maxima of the fitness function in this problem. In this approach (as already mentioned) the size of the circuit is much more controlled, however it is required that the initial estimate of the maximal size of the circuit appropriately over-estimates the minimal length (or the expected minimal length) of the circuit. This is important as it allows to design the given function with various costs and thus to obtain different results close to the minimum. Thus, the search space has to be restricted around the expected minimum as close as possible. Moreover, as the GA converges towards a local or global minimum (maximum fitness) the introduction of the new individuals should be in the problem space close to the global optimum. This way, there will be a global convergence in the desired region of the problem space.

4.1 Symbolic synthesis - single output functions

The GA was used to search three and four qubit-single output circuits for fully or partially defined functions. Some of the benchmarks have already been introduced in Section 3 and they all are incompletely specified functions. Table 14 shows some benchmark functions used in the discussed experimentations and their qubit numbers.

Table 14. GA benchmark functions

$f_1 = [-,-,0,1,0,1,1,0,0,0,-,-,0,1,1,0]$	4
$f_2 = [0,1,-,-,-,-,1,0]$	3
$f_3 = [0,0,-,-,0,1,V_0,-]$	3
$f_4 = [0,-,0,-,V_0,1,-,1]$	3
$f_5 = [-,V_0,-,1,-,1,V_0,-]$	3
$f_6 = [0,0,0,1,0,1,1,1]$	3
$f_7 = [0,-,0,M_{0,1},M_{0,1},1,-,1]$	3

For the discussion of the results and synthesis four of these benchmarks have been selected. In particular, functions f_4, f_5, f_6 and f_7 have been analyzed and studied.

Example 4.1.1
The first function was $f(a,b,c,d) = M_f = [0,-,0,-,V_0,1,-,1]$ with the least significant qubit being the output. As already introduced the V_0 operator represents the probabilistic behavior of this function for a given input. Its realization is shown in Figure 15, with top qubit being constant 0 $|0\rangle$ and bottom qubit being the output. Because the measured qubit can be entangled with the rest of the circuit, the output qubit can present the entangled behavior. One observation can be made about the function realization from Figure 15. This circuit maps the desired outputs (specification) to: $\{|0000\rangle \rightarrow |0000\rangle, |0010\rangle \rightarrow |0010\rangle, |0100\rangle \rightarrow \frac{1}{2}(|0100\rangle) + \frac{1}{2}(|0101\rangle), |0101\rangle \rightarrow |0001\rangle, |0111\rangle \rightarrow |0011\rangle\}$. This illustrates an exact matching of the circuit mapping to the desired function. With respect to the single-output function learning method introduced in this paper this circuit can be written as: $[0,1,0,1,V_0,1,V_0,1]$ which covers the quantum cares with 100% accuracy.

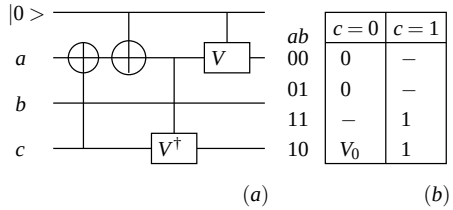
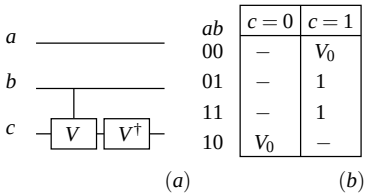


Fig. 15. Benchmark function 1: a - the learned circuit, b - the initial specification

Example 4.1.2
The next benchmark is another partially specified function, shown in Figure 16. Similarly to the previous example, this function is defined over a subset of desired mappings. Circuit from the Figure 16 generates the desired outputs such as: $\{|001\rangle \rightarrow \frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |110\rangle) + \frac{1}{2}(|101\rangle + |111\rangle), |011\rangle \rightarrow \frac{1}{2}(|101\rangle) + \frac{1}{2}(|111\rangle), |111\rangle \rightarrow \frac{1}{2}(|001\rangle) + \frac{1}{2}(|011\rangle), |100\rangle \rightarrow \frac{1}{2}(|000\rangle + |010\rangle + |100\rangle + |110\rangle) + \frac{1}{2}(|001\rangle + |011\rangle)\}$. As can be seen this function uses superposition to generate the correct result.

Interestingly, adding two gates: Controlled-$V^\dagger$ on wires b and c at the beginning of the circuit and Controlled-V right after the first controlled-Not on the a and b wires, will generate partially deterministic and partially probabilistic behavior. Again, it is possible to describe this function using quantum symbolic values: $[V_1,V_0,0,1,V_1,V_0,0,1]$. Also observe that we allow the change from V_0 to V_1 in the case of the input states $|100\rangle$ and $|101\rangle$. This is both because under normal standard projective measurement it is not possible to distinguish easily between the two states (both states will generate a 0 with 50% probability and 1 with 50% probabil-



ab	c = 0	c = 1
00	–	V_0
01	–	1
11	–	1
10	V_0	–

Fig. 16. Benchmark function 2: a - the learned circuit, b - the initial specification

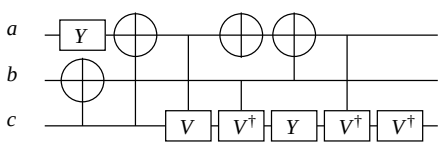
ity) as well as because $V|1\rangle = V^\dagger|0\rangle$. However depending on the specification of the behavior the solution can have multiple undesired outputs.

Example 4.1.3
The last result is for the 3-qubit majority function [5]

$$f = \{0,0,0,1,0,1,1,1\} \tag{6}$$

.

The function is shown on Figure 17 and is a pure permutative matrix.



ab	c = 0	c = 1
00	0	0
01	0	1
11	1	1
10	0	1

Fig. 17. Function Majority 3.

In this case the analysis is a little more complicated

$$[VV^\dagger, V^\dagger V^\dagger, VV^\dagger, VV^\dagger, V^\dagger V^\dagger, VV^\dagger V^\dagger V^\dagger, V^\dagger V^\dagger, VV^\dagger] \tag{7}$$

however again the correct result is obtained. Observe that the Pauli Y rotation operator was used by the Genetic Algorithm and that this solution is not minimal.

From the point of view of the synthesis costs it is valid to ask if the concepts of superposition and entanglement can be useful for learning and logic synthesis. In other words, can the given function be designed cheaper using superposition and entanglement? So far, the research in quantum logic synthesis is on the usage of a particular set of gates for minimization, possibly resulting in entanglement, but it is possible that logic synthesis should intentionally use the entanglement, superposition and measurement to synthesize the desired function, either measurement

independent of measurement dependent. This problem remains an open research area.

The GA synthesis algorithm itself used in this work is similar to our previous work in this area [28, 25] however particular settings allowed to generate circuits from certain subspaces. For example, when one wishes to synthesize circuits using the presented method, the set of declared component gates must contain only those gates that the user intends to be used. Thus, it is simple to influence the behavior of the GA in this direction. More interesting is the problem of the error/fitness function. For example assume that the GA is to synthesize circuit that has to have at least one state in superposition. This will generate a state that is in superposition. However, now assume that the designer wants to synthesize a two qubit output state transition such as $|00\rangle \rightarrow \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. If the GA is using only the probabilities of outcome, the specification in the amplitudes of observation such as above could easily synthesize the output using simple superposition. Thus for higher dimensionality of quantum circuits, particular measurement operator must be generated allowing to distinguish the desired states.

5 Conclusions

We showed a new approach to machine learning; i.e. learning quantum circuits from partially specified examples with symbolic quantum states. This has application in robotics [8, 9]. The behavior is specified by a truth table with don't cares and symbolic quantum states such as $V_0 = V|0\rangle$ and $V_1 = V|1\rangle$ which lead to known probabilities of observation of the output under certain measurement operators[25]. A circuit matching the symbolic cares of the specification is found. This circuit has both deterministic and probabilistic behaviors when one of its qubits is measured. Such behaviors have been designed for robots and observed on several small mobile and humanoid robots[8, 9]. This paper introduced also the concept of a quantum function with binary inputs and a single (quantum symbolic) binary output. We formulate the synthesis problem for such functions and solve this problem using the genetic algorithm and exhaustive search. Quantum truth tables can be represented as standard Karnaugh maps with symbolic quantum states as their outputs. We called them Quantum Karnaugh Maps. This new representation has a didactic value because it links the quantum concepts to the binary synthesis concepts such as minterms, symmetry, unatness, implicants and other that people are familiar with while using K-maps. This representation, similar to maps for binary input multiple-valued output functions, allows to create new quantum circuits from K-maps. These methods are more knowledge-based than the GA or exhaustive search. We created for instance methods based on encircling large groups of symbols 1, V_0 and V_1

similarly as it is done in hand methods for ESOP synthesis.

Finally, the future work includes:

1. extension of the synthesis process to multi-output reversible, and truly quantum functions, and
2. observation of physical robots with their controllers simulated on a quantum simulator.

In particular, we are investigating the quantum robot controllers with respect to social behavior and the automatic generation of social behaviors.

References

[1] B. Zupan, M. Bohanec, I. Bratko, and J. Demšar, "Machine learning by function decomposition," in *Proc. 14th International Conference on Machine Learning*. Morgan Kaufmann, 1997, pp. 421–429. [Online]. Available: citeseer.ist.psu.edu/article/zupan97machine.html

[2] C. M. Files and M. A. Perkowski, "An error reducing approach to machine learning using multi-valued functional decomposition," in *Proceedings of the International Symposium on Multiple-Valued Logic 1998*, 1998, pp. 167–172.

[3] S. Grygiel, *Decomposition of Relations as a new approach to constructive Induction in Machine Learning and Data Mining, Ph.D. dissertation*. Portland State University, 2000.

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[5] D. M. Miller, D. Maslov, and G. W. Dueck, "Synthesis of quantum multiple-valued circuits," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 12, no. 5-6, pp. 431–450, 2006.

[6] A. Muthukrishnan and C. R. Stroud, "Multivalued logic gates for quantum computation," *Phys. Rev. A*, vol. 62, no. 5, p. 052309, Oct 2000.

[7] J. Gruska, *Quantum computing*. Osborne/McGraw-Hill,U.S., 1999.

[8] A. Raghuvanshi, Y. Fan, M. Woyke, A. Kumar, and M. Perkowski, "Quantum robots for teenagers," in *Proceedings of the International Symposium on Multiple-Valued Logic 2007*, 2007.

[9] M. Lukac and M. Perkowski, "Quantum mechanical model of emotional robot behaviors," in *Proceedings of the International Symposium on Multiple-Valued Logic 2007*, 2007.

[10] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and W. H., "Elementary gates for quantum computation," *The American Physical Society*, vol. 5, pp. 3457–3467, 1995.

[11] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and systems*, vol. 25, no. 9, pp. 1652–1663, 2006.

[12] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, C. H. Yu, K. Chung, H. Jee, B.-G. Kim, and Y.-D. Kim, "Evolutionary approach to quantum and reversible circuits

synthesis," in *Artificial Intelligence in Logic Design*. Kluwer Academic Publisher, 2004, pp. 201 – 257.

[13] L. Grover, "A fast quantum mechanical algorithm for database search," 1996. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/9605043

[14] P. Benioff, "Quantum robots and environments," *Physical review. A*, vol. 55, no. 2, pp. 893–904, 1998.

[15] ——, "Quantum mechanical hamiltonian models of turing machines," *Journal of Statistical Physics*, vol. 29, no. 3, pp. 515–546, 1982.

[16] ——, "Space searches with a quantum robot," *AMS CONTEMPORARY MATH SERIES*, vol. 305, 2002. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0003006

[17] ——, "Some foundational aspects of quantum computers and quantum robots," *Superlattices and microstructures*, vol. 23, no. 3-4, pp. 407–417, 1998.

[18] D. Dong, C. Chen, C. Zhang, and Z. Chen, "Quantum robot: structure, algorithms and applications," *Robotica*, vol. 24, no. 4, pp. 513–521, 2006.

[19] M. Hunziker, D. Meyer, J. Park, J. Pommersheim, and M. Rothstein, "The geometry of quantum learning," 2003. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0309059

[20] D. Curtis and D. Meyer, "Towards quantum template matching," pp. 134–141, 2004.

[21] N. H. Bshouty and J. C. Jackson, "Learning dnf over the uniform distribution using a quantum example oracle," in *COLT '95: Proceedings of the eighth annual conference on Computational learning theory*. New York, NY, USA: ACM Press, 1995, pp. 118–127.

[22] D. Ventura and T. Martinez, "A Quantum Computational Learning Algorithm," *ArXiv Quantum Physics e-prints*, July 1998.

[23] D. Ventura, "Implementing competitive learning in a quantum system," in *International Joint Conference on Neural Networks, IJCNN*, vol. 1, 1999, pp. 462 – 466.

[24] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Provably optimal reversible quantum logic synthesis via symbolic reachability analysis," in *Proceedings of DAC*, 2004.

[25] M. Lukac and M. Perkowski, "Combining evolutionary and exhaustive search to find the least expensive quantum circuits," in *Proceedings of ULSI 2005*, 2005.

[26] ——, "Using exhaustive search for the discovery of a new family of optimum universal permutative binary quantum gates," in *Proceedings of International Workshop on Logic & Synthesis, Poster Session*, 2005.

[27] T. Sasao and J. Butler, "The eigenfunction of the reed-muller transform," in *Proceedings of the RM*, 2007, pp. 31–37.

[28] M. Lukac, M. Pivtoraiko, A. Mishchenko, and M. Perkowski, "Automated synthesis of generalized reversible cascades using genetic algorithms," in *Proceedings of Fifth Intern. Workshop on Boolean Problems*, 2002, pp. 33–45.