Don’t Do Things You Can’t Undo: Reversibility Models for Generating Safe Behaviours

Maarja Kruusmaa, Yuri Gavshin, Adam Eppendahl

***Abstract*—We argue that an ability to determine the reversibility of actions allows a robot to identify safe behaviors autonomously. We introduce a notion of reversibility model and give a definition of model refinement. We implement this on a real robot and observe that, when a reversibility model is refined by the addition of proximity sensors, obstacle avoidance emerges as a side-effect of avoiding irreversible actions. We interpret this as evidence of a deep connection between reversibility and safe behaviour. We also observe that, on the real robot, reversiblities are learned as efficiently as a dedicated reward function. We conclude that reversibility identification may provide an abstract and yet practical method of generating a variety of safe behaviours.**

I. Introduction

This paper is concerned with a robot’s ability to undo its actions. We suggest that reversibility, being a necessary condition of controllability, is a fundamental concept when programming robots to behave safely and reliably. We ask if this principle can be used to govern the operation of the robot, and to generate useful behaviour on a real robot and in real time.

We speculate that the most undesirable actions in the real world (for example, those that damage the robot or the environment) are characterized by irreversibility. Thus, instead of teaching the robot specific routines such as avoiding collisions, avoiding falls, etc., we teach the robot a more general principle of avoiding irreversible actions. In other words, instead of telling the robot *what* should not be done, we try to tell it *why* it should not be done. For example, falling down the stairs is not good because the robot does not know how to climb back or pushing the door closed is not good because it does not have knowledge of how to open it.

In this paper, we state the problem of learning a reversibility model. The reversibility model captures the robots knowledge of state-action pairs that are reversible and the ways of reversing them. We go on to demonstrate how this reversibility model can be acquired and used to generate new behaviours. In our experiments we show that by suppressing irreversible actions the robot will develop obstacle avoidance behaviour. As a developmental system, the efficiency of this approach is comparable to reinforcement learning. The difference here is that the reinforcement learning algorithm requires a signal that identifies collisions and labels them as the undesirable, while the reversibility algorithm identifies the undesirable behaviours by their abstract properties and this just happens to result in collision avoidance. Thus we see a safe concrete behaviour emerging autonomously from a very abstract universal principle.

An enormous amount of robot literature is concerned with algorithms for avoiding collisions as this is considered an essential ability for mobile robots. In this literature, the goal of avoiding collisions is explicitly stated [1], while the solution may be coded for by hand or obtained indirectly using learning algorithms [2,3]. Efficient navigation can be learned, for example, by using genetic algorithms [4], adaptive fitness functions [5], neural networks [6] or Q-learning [7]. In [8], navigation behaviours are derived by classifying random sensor data. Our approach is different in that reliable navigation emerges from an abstract rule. The rule is not grounded in a specific sensor-motor semantics that explicitly identifies collisions. The resulting developmental system is insensitive to sensor permutations and inversions. The practical benefit of this is that the code can be written without knowing the location or polarity of sensors and actuators.

The idea of generating behaviours top-down from abstract principles is an emerging theme in parts of the autonomous robotics community. In developmental robotics, for example, relatively abstract emotional and motivational mechanisms are used to derive behaviours that facilitate social interaction
[9] [10]. Kaplan and Odeyer show that a number of basic visual behaviours can emerge from abstract motivational principles based on prediction errors [11]. The general idea behind the approach is to identify principles that can be expressed without reference to the ground meaning of sensor-motor values. Code based on such principles should function reliably in a broad range of environments and on different robots or on different parts of the same robot. Our principle of avoiding irreversible actions provides just one example of such an abstract ungrounded principle.

This work was supported in part by the Estonian Science Foundation grant No. 6765.

Maarja Kruusmaa, corresponding author is with the Tartu University Institute of Technology, Nooruse 1, 50411 Tartu, Estonia. Phone: (+372) 51 83074, fax: (+372) 7374900, e-mail: maarja.kruusmaa@ut.ee

Yuri Gavshin is with Dept. of Computer Science, Tartu University, Liivi 2, Tartu, Estonia, e-mail: yuri.gavshin@ut.ee

Adam Eppendahl is with the Department of Design and Manufacturing, University of Malaya, Kuala Lumpur, Malaysia, e-mail: a.eppendahl@mac.com

In the following section we present these ideas about reversibility in a more formal manner. After that, we describe an experimental set-up for a Khepera mini-robot to test the reversibility principle. We describe the experimental design and present the results. In the last section we discuss the results, draw conclusions and envision possible directions for future work.

II. Reversibility Models

A reversibility model tells the robot which actions are reversible and how to reverse them if they are. In a fixed, known, exact, deterministic world, modelled by a graph G of states and actions, an action from state s to state s' is reversible if there is an action back from s' to s. If we admit sequences of actions, by taking $G = Path(G_0)$ (the graph of paths over G_0), where G_0 is some graph of atomic actions, then finding reversibilities in G is equivalent to finding loops in G_0, a standard problem in graph theory.

This is all very well for playing Sokoban, but real robots face a non-deterministic, inexact, partially known and changing world. Therefore, we model non-determinism using labelled transition systems, we allow inexactness with a metric on the space of states, and we define a reversibility model pragmatically to be a set of expected reversibilities that may grow or shrink as the robot gains experience.

In addition, the robot may itself be changing as it learns, develops or reconfigures, and this is what interests us most. In this paper we consider one such change, the addition of sensors, and introduce a notion of refinement that captures the relationship between the robot's world before and after the change. In the learning experiments we describe, a reversibility model for an unrefined world is adapted to a refined world (with the interesting side-effect of producing obstacle avoidance behaviour).

Suppose we have a set of states given by vectors of sensor values and a set A of actions given by vectors of motor commands. If we view the states as the nodes in a graph and the actions as labels, the robot's body and environment determine a labelled transition system which we refer to as the robot's *world.* A labelled transition system is a standard structure for modelling non-determinitstic systems and consists of a directed graph with edges, called transitions, labelled by actions. When the result of an action a in state s is not wholly determined by the robot, multiple transitions from s are labelled with the same action a and it is the world that determines which transition actually happens.

A *reversibility* for a world W is a state-action pair (s,a), together with a state-action pair $(s',\bar{a})$. A reversibility may or may not hold, in a mathematical sense or in a physical sense. Generally speaking, $\bar{a}$ is expected to produce a transition from s' to s, assuming a produces a transition from s to s' to s' in W. Because of the non-determinism, even given a perfectly known world W, there are different ways to define 'holding'. A reversibility $((s,a),(s',\bar{a}))$ may *hold weakly* if there exists in W a transition from s to s' labelled a and a transition from s' to s labelled $\bar{a}$. Or, it may *hold strongly* if there exists a transition from s to s' labelled a and every transition from s' labelled $\bar{a}$, and at least one, leads to s. In our implementation, we use the strong definition. Also, the action $\bar{a}$ is expected to work for any state x with $d(x,s') < \varepsilon$ and is only expected to produce a transition back to a state y when $d(y,s) < \varepsilon$, where d is a metric on states.

A *reversibility model* for a world W is a set of reversibilities for W that are expected to hold. In practice, a reversibility model could be given in advance, communicated to the robot, learned empirically, deduced from knowledge about the world, or obtained in some other way. In the experiments described here, the robot is given a model for one world and uses this to learn a model for a refined world.

A *refinement (of states)* from a world W to a world W' is a pair of functions from the states and transitions of W' to those of W that respects the graph structure and labelling and is surjective on states. In other words, every state in W is the image of one or more states in W', which 'refine' the state in W, and the action on an edge in W' is given by the action on the edge it is sent to in W.

For any reversibility model R for a world W and for any refinement from W to W', with state function p, there is a refined set of reversibilites R' on W' defined by

$$R' = \{((s,a),(s',\bar{a})) \mid ((p(s),a),(p(s'),\bar{a})) \in R\}.$$

To obtain a reversiblity model for the new world W' we may form R' and then remove any pairs that fail in the refined world. An important aspect of this procedure is that 'it gives the robot something to do': the original model R provides a specific list of actions together with the circumstances in which they should be tried.

The kind of refinement we have in mind is produced by extending a robot's sensor vector. Suppose we have a world with states given by pairs of wheel counter values (w_1,w_2) and actions given by pairs of wheel displacement commands (m_1,m_2). Assuming the robot is able to control its own wheels, this world is fairly deterministic, all actions are reversible and a good reversibility model R is given by taking $\bar{a} = (-m_1,-m_2)$ when $a = (m_1,m_2)$ (for any s and s').

Now suppose we include one proximity value (say, the front sensor) in the state vector (w_1,w_2,d_1). Assuming the new sensor does not effect the robot's environment, we obtain a refinement of the original world. The state function p is the projection

$$p(w_1, w_2, d_1) = (w_1, w_2).$$

When the simple model R described above is refined according to this new world some of the refined reversibilities hold and some do not. In our experiments, the robot tests these refined reversibilities to discover which hold.

The interesting point here is that the ones that fail generally correspond to collisions of some sort. Consider the following four cases (in which wheel counts and proximities are given, without loss of generality, in comparable units). (1) The robot does not touch anything: we obtain, say, the successful reversiblity

$$(((0, 0, 15), (10, 10)), ((10, 10, 5), (-10, -10)),$$

where the robot approaches and retreats from an object without touching it. (2) The robot touches an object and the object slides: we obtain a failed reversibility, say

$$(((0, 0, 8), (10, 10)), ((10, 10, 0), (-10, -10)),$$

where the robot runs into an object, pushing it 2 units forward, then retreats, and then finds that, while its wheel encoders are back to 0 as expected, its proximity sensor now reads 10 instead of the original 8. (3) The robot touches an object and its wheels slide: from the robots point of view, this is identical to case 3. (4) The robot touches an object and jams: if motor commands time-out and report success, adjusting the wheel encoder counts as necessary, then this case is again identical to case 3 (and may be thought of as a kind of internal sliding).

Not only does the robot discover that it is 'bad' to push things—without ever knowing what pushing is!—but the refined state allows the robot to distinguish those cases in which 'bad things happen' from those in which they do not. Once the robot learns a reversibility model, it may use the model to censor its actions. Because of the non-determinism, we have a growing choice of definitions. A state-action pair (s, a) is weakly (strongly) reversible in world W, if there is a reversibility $((s,a),(s',\overline{a}))$ that holds weakly (strongly) in W for every s' that can be reached from s by an a transition. Alternatively, we could ask for just one such s'. In our experiments, we use, in effect, the strong definition, but because we pretend the world is deterministic by ignoring s (by taking $\varepsilon' = \infty$), there is no real difference.

Note that it is our method of creating a reversibility model out of R' (by pruning) that creates a 'pushing is bad' model. Alternatively, when a reversibility $((s,a),(s',\overline{a}))$ in R' fails, we could try replacing the action $\overline{a}$ instead of throwing out the reversibility. For example, we could construct the world W $W'' = Path(W')$. The transitions in $Path(W')$ are paths of transitions in W' labelled by sequences of actions from W'. The world W' embeds in W'', along with R', but now we have sequences of actions to play with. In the object pushing example, a sequence b of actions might cause the robot to go behind an object, push it back 2 units, and then return to its original place in front of the object, so that

$$(((0, 0, 8), (10, 10)), ((10, 10, 0), b),$$

holds in W''. Or we could form W'' by add a gripping action and simply drag the object back 2 units.

III. Experiments

This section describes two learning algorithms. One learns which reversibilities hold or fail when the robot's world is refined by the addition of eight proximity sensors. The other one is a reinforcement algorithm with a reward function that punishes collisions. The learning performance of the two algorithms are compared in two test environments, an easy one and a harder one, and with two sets of actions, 1D and 2D.

A. Implementation Details

An action $a = (m_1, m_2)$ consists of a pair, left and right, of motor displacement commands expressed in the native wheel decoder units. A discrete set of actions is used in the experiments:

$a_1 = (100,100)$ short step forward,
$a_2 = (300,300)$ long step forward,
$a_3 = (-100,-100)$ short step backward,
$a_4 = (-300,-300)$ long step backward,
$a_5 = (100,-100)$ rotate clockwise, and
$a_6 = (-100,100)$ rotate counterclockwise.

In the 1D experiments, we take $\mathbf{A} = \{a_1, a_2, a_3, a_4\}$. These actions cause the robot to move back and forth in a straight line. In the 2D experiments, we include the turning actions, $\mathbf{A} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$.

We provide the robot with the initial reversibility model

$$\{((x,a_1),(x+(100,100),a_3)),$$
$$((y,a_2),(y+(300,300),a_4)),$$
$$((z,a_5),(z+(100,-100),a_6))\},$$

where x, y and z are any states (w_1,w_2), consisting of a pair of wheel counter values. Because we have fixed things so that wheel commands always succeed, the reversibilities in this model always hold. We then use (in effect) a refinement function p, the projection from the set of states $(w_1,w_2,d_1,d_2,d_3,d_4,d_5,d_6,d_7,d_8)$, which include eight proximity values, to the original set of states without the proximity values, to induce a new set of refined reversibilities from the original set. The new set contains, for example,

$$((s,a_1),(s',a_3)) =$$
$$(((w_1,w_2,d_1,d_2,d_3,d_4,d_5,d_6,d_7,d_8),a_1),$$
$$((w_1+100,w_2+100,d'_1,d'_2,d'_3,d'_4,d'_5,d'_6,d'_7,d'_8),a_3),$$

for any w_i, d_i and d'_i. The learning algorithm then tests these to see which hold and which fail.

For our definition of 'near', we use the Hamming metric defined by $d(s,s') = \sum_{i=1}^{2} |w_i - w'_i| + \sum_{i=1}^{8} |d_i - d'_i|$, (but because our wheel commands always succeed, and the original model is correct, the wheel value part of this is always 0.) In our set-up we know that if the robot drives against an object, the wheels will jam (instead of sliding) so we can detect collisions by watching for wheel controller time-outs.

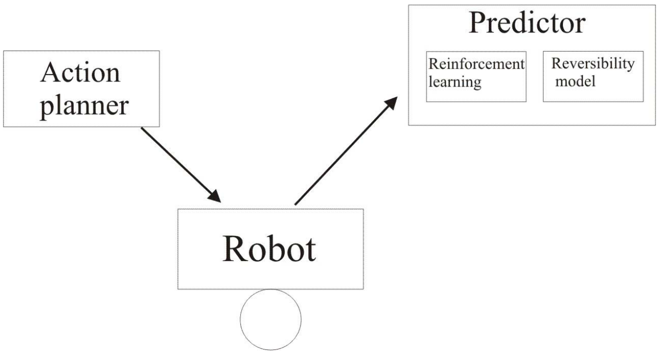

Fig. 1. Experimental setup.

1) Robot motion. The Khepera runs in a real, physical environment with motions that test the pairs of the refined reversibility model. The robot moves according to the following algorithm:

```
1. Record current state $s_i = (w_1, w_2, d_1, \dots, d_8)$.
2. Choose an arbitrary reversibility from $\mathbf{R'}$
   and execute the forward action as $a_i$.
3. Record the state $s_{i+1} = (w'_1, w'_2, d'_1, \dots, d'_8)$
4. Execute the reverse action as $a_{i+1}$.
5. Add 2 to $i$.
```

So the robot performs a random action, then it's reverse action, and then another random action, etc.

2) Learning the reversibility model. As the robot moves about, it notes how well the reversibilities hold using the Hamming metric.

```
For each forward action $a_i$, calculate and
store $d(s_i, s_{i+2})$.
```

For the purposes of comparison with the reinforcement algorithm, the model is also used to predict which actions will be successfully reversed. When a failure is predicted, we note whether there is a collision during the action. So we are judging the reversibility model not by what it is meant to be learning, but by how well this happens to predict collisions.

```
1. Get the current state $s_i$ and the intended
   action $a_i$
2. From memory, choose a state-action pair
   $(s_k, a_k)$ that minimizes $d(s_k, s_i)$.
1. If we do not have $d(s_k, s_i) < \delta$ predict
   randomly. Otherwise, predict a collision
   unless $d(s_k, s_{k+1}) < \varepsilon$.
2. While executing the command $a_i$ check if
   there is a collision. Store the predicted
   and the actual outcome.
```

3) Reinforcement learning. Reinforcement learning algorithms are commonly used for learning in mobile robotics. The goal here is to implement a simple version for collision avoidance to compare the ungrounded reversibility method to a standard, grounded method. We have therefore implemented the reinforcement learning algorithm so that the robot is operating under similar conditions. First, the algorithm does not have a terminal state, so collision avoidance is considered to be a continuous task of reward maximization. Second, the current version of the reversibility policy is concerned only with immediate actions and reverse actions and does not work along the history of action sequences. Therefore we have also implemented the reinforcement algorithm to be concerned only with immediate rewards, thus with discount rate $\gamma = 0$. The initial value of the action value function is $Q(s_i, a_i) = 0$. The reward signal is defined by checking for collisions.

$$r = \begin{cases} |w_1| + |w_2| / 100, \text{ if there is no collision} \\ -5, \text{ if there is a collision} \end{cases}$$

Thus a successful action is rewarded more if it moves the robot for greater distance and an unsuccessful action is strongly penalized.

Note that the reinforcement learning algorithm directly checks for collisions to calculate the reward, while the algorithm learning the reversibility model only aims at predicting if the robot can return to the initial state.

The reinforcement learning algorithm is the following:

```
1. Get the current state $s_i$ and the intended
   action $a_i$
2. If the current value of the action value
   function $Q(s_i, a_i) > 0$, predict no
   collision. Predict a collision otherwise.
   If $Q(s_i, a_i) = 0$ make a random prediction.
3. After executing $a_i$ get the reward signal
   $r$.
4. Update the action value function
   $Q(s_i, a_i) \leftarrow \alpha r + Q(s_i, a_i)$ where the learning
   rate $\alpha = 0.1$.
5. While executing $a_i$, check for collisions.
   Store the predicted and the real outcome.
```

B. Test environments

In the following experiments we compared the learning of reversibility models to the learning of a reward function that discourages collisions. To find out how sensitive the learning algorithms are to environmental conditions, the tests are conducted in two environments. Environment II is smaller than Environment I, the collisions with walls at each move are more probable. Also the walls are not perpendicular as in Environment I. In both environments the algorithms are implemented in 1D and 2D to get the estimate of their scalability. In 1D the robot can move only back and forth, while in 2D turning is permitted.

IV. RESULTS

As it was specified in the previous section, the robot operates by executing reversed and random actions. The reversed actions are determined according to the initial reversibility model. The goal of the learning algorithms is to observe and learn to predict the outcomes of the actions. These predictions are then compared to the real outcome of the action (determined by detecting collisions) and the success rate of both of the policies is recorded. The tests are conducted in 1D environment where only the back and forth moves are permitted and in 2D where turning actions are also included.

Fig. 2 – Fig. 5 represent the test results in both test environments. All charts represent the average correctness of predictions for every successive 100 actions for both prediction policies.

From Fig. 2 and Fig. 4 it can be seen that in both environments, the robot learns very fast to avoid collisions in 1D, the rate of successful predictions reaches 80%-90% during the first 200-300 steps and the learning problem is equally trivial for both learning algorithms.

In 2D, the turning actions make the learning problem more complicated, both algorithms reaching similar accuracy around 1900-2100 steps. Environment I is more "friendly" for the robot while in Environment II the collisions are much more likely to occur.

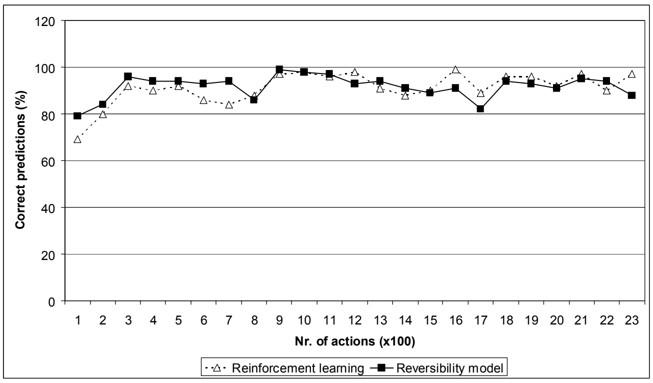

Fig. 2. Test results in environment I, in 1D

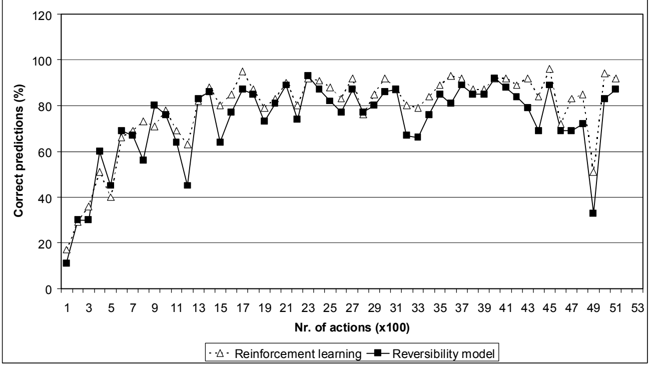

Fig. 3. Test results in environment I, in 2D

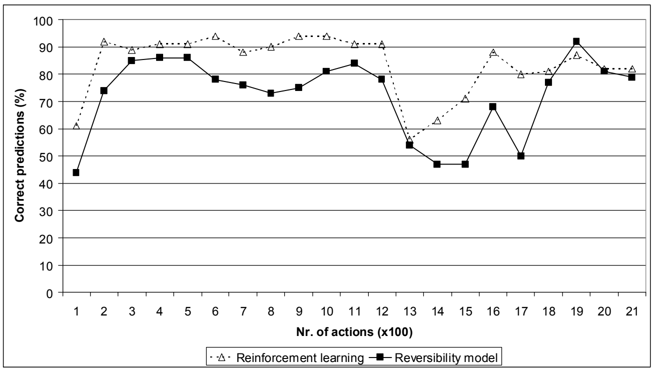

Fig. 4. Test results in environment II, in 1D

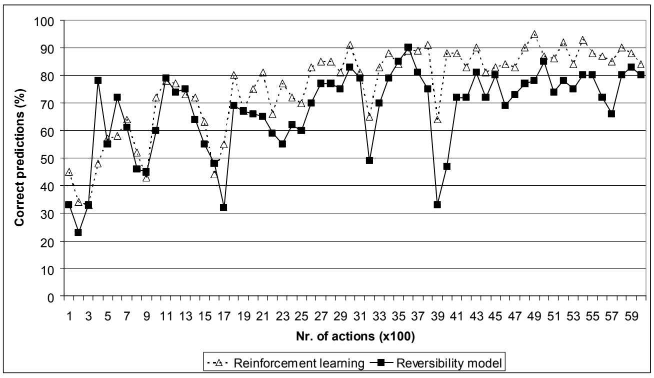

Fig. 5. Test results in environment II, in 2D

Also, during the tests in Environment II the wheels occasionally got stuck on the uneven surface. These occasions can be seen on the charts around 1400-1700 steps in 1D (Fig.4) and 3700-4100 in 2D (Fig.5) with the sharp downward peaks of the prediction rates. It appears that reinforcement learning adapts and recovers better in these occasions. However, this is caused more by the method we use to determine the prediction rate than the failure of the method of learning the reversibility model. For the robot with the blocked wheels the reversibility of actions is perfect, since the robot certainly ends up in the same state it starts from. However, if this success of the prediction is judged by whether or not the wheels jam, the prediction is wrong.

The general conclusion drawn from the experimental results is that both methods learn to predict collisions with comparable speed. The reinforcement learning algorithm uses a reward signal determined by the outcome of an action whereas the reversibility policy uses the reversibility measure to refine the initial reversibility model.

The reversibility model is learned with a roughly similar speed in Environment I whereas in Environment II reinforcement learning policy performs slightly better. Also, upscaling from 1D to 2D environment is equally efficient for both approaches.

The most important comment is that the aim of this research is not to develop a competitive learning technique for collision avoidance but rather to introduce a new general principle that can be used in combination with existing methods. Our method does not learn how to avoid obstacles, it learns how to repeat its actions. An important result of the experiments is that a useful behaviour derived from a general abstract principle can be learned as fast as by an established method for learning collision avoidance.

V. CONCLUSIONS

This paper introduces the concept of reversibility for learning robots. We show that reversibility models can be used to learn a useful new behaviour. The experiments verify the performance of the reversibility method against a well-established method of learning commonly used in robotics. The results show that both of the methods converge to obstacle avoidance behaviour.

The most general conclusion drawn from the experimental results is that the efficiency of the policy of reversibility is comparable to reinforcement learning. Both methods learn more or less equally, converging to satisfactory performance. The basic difference of these methods is that the reinforcement learning algorithm uses a reward signal explicitly designed to make the robot avoid obstacles. The policy we introduce, uses a reversibility measure to learn a reversibility model, and yet the robot learns the useful behaviour of collision avoidance.

Based on these experimental results we speculate that the concept of reversibility could generate a variety of useful behaviours depending on the properties of the environment. We surmise, that for example for a robot, placed initially close to an object or wall, the robot using reversibility models might discover behaviours like "do not leave the territory" or "stay in the vicinity of guidlines". Our future experiments are planned to check this hypothesis and find more evidence concerning the robustness of the principle.

Another hypothesis we are planning to test is whether learning algorithms can be accelerated by using reversibility models. Generally, learning algorithms converge to a stable behaviour by repeating actions that lead from one state to another. It is not explained, however, how the robot gets back to the state it wants to repeat. Knowing the reversibility model, it may be easier to guide the learning algorithm to faster convergence.

We also suggest that reversibility models could be used in combination with formal reasoning methods, such as task or path planning, where the plans can be checked for reversibility. For mobile robots such a reversibility check could, for example, guarantee safe homing or safe exploration. We suggest that the concepts introduced in this paper may provide handy and simple guidelines for building safe and reliable robots.

REFERENCES

[1] J. Borenstein, Y.Koren, Real-time obstacle avoidance for fast mobile robots" in *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 19, No. 5, Sept./Oct., pp. 1179-1187.

[2] R.Arkin, "Behavior-based robotsics," MIT Press: Cambridge, MA

[3] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. Brooks and P. Maes, ed.s, Artificial Life IV, pages 190--197. MIT Press/Bradford Books, 1994..

[4] D.Bajaj, M. Ang, Jr., "An incremental approach in evolving robot behavior," in *Proceedings of the Sixth International Conference on Control, Automation, Robotics and Vision (ICARCV'2000),*5-8 Dec 2000, Singapore.

[5] E. Uchibe, M.Yanase, and M. Asada. "Behavior generation for a mobile robot based on the adaptive fitness function," in *Robotics and Autonomous Systems,* Vol.40, pp.69-77, 2002.

[6] J. Blynel, D. Floreano, "Exploring the T-Maze: Evolving Learning-Like Robot Behaviors using CTRNNs," In *Applications of Evolutionary Computing,* 2003.

[7] G.-S. Yang, E.-K. Chen, C.-W. An, "Mobile robot navigation using neural Q-learning", in *Proc. Of Int. Conf of Machine Learning and Cybernetics,* Vol.1, pp. 48-52, 2004.

[8] E. Simonin, J. Diard, P. Bassiere, "Learning Bayesian models of sensorimotor interaction: from random exploration toward the discovery of new behaviors," In *Proc. Of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 1226-12231.

[9] C. Breazeal, Designing Sociable Robots, Cambridge:MIT Press. 2002

[10] L. Moshkina, R.C.Arkin, "On TAMEing Robots", *Proc. IEEE International Conference on Systems, Man and Cybernetics,* Vol. 4, pp. 3949-3959. Oct. 2003.

[11] F. Kaplan, P.Y. Oudeyer, „Motivational principles for visual know-how development," In *P,roceedings of the Third International Workshop on Epigenetic Robotic,s* 2003.

[12] R.S.Sutton, A.G.Barto, Reinforcement Learning. An Introduction, MIT Press, Cambridge, MA, 1998, A Bradford Book