Document downloaded from:

http://hdl.handle.net/10251/37392

This paper must be cited as:

Insa Cabrera, J.; Dowe, DL.; Hernández Orallo, J. (2011). Evaluating a reinforcement learning algorithm with a general intelligence test. En Advances in Artificial Intelligence. Springer Verlag (Germany). 7023:1-11. doi:10.1007/978-3-642-25274-7_1.

The final publication is available at

http://link.springer.com/chapter/10.1007/978-3-642-25274-7_1

Copyright Springer Verlag (Germany)

Evaluating a reinforcement learning algorithm with a general intelligence test

Javier Insa-Cabrera[1] David L. Dowe[2] José Hernández-Orallo[1]

[1] DSIC, Universitat Politècnica de València, Spain. {jinsa, jorallo}@dsic.upv.es
[2] Clayton School of Information Technology, Monash University, Australia. david.dowe@monash.edu

Abstract. In this paper we apply the recent notion of anytime universal intelligence tests to the evaluation of a popular reinforcement learning algorithm, Q-learning. We show that a general approach to intelligence evaluation of AI algorithms is feasible. This top-down (theory-derived) approach is based on a generation of environments under a Solomonoff universal distribution instead of using a pre-defined set of specific tasks, such as mazes, problem repositories, etc. This first application of a general intelligence test to a reinforcement learning algorithm brings us to the issue of task-specific vs. general AI agents. This, in turn, suggests new avenues for AI agent evaluation and AI competitions, and also conveys some further insights about the performance of specific algorithms.

1 Introduction

In order to evaluate progress in AI, intelligence and performance tests are crucial. We know about many AI competitions held in many different realms (learning, planning, robotics, games, ...). Most of them, however, are just constructed as a set of specific tasks. While many of these competitions modify and extend the set of tasks each year in order to cover a broader view of the field and avoid competition-specialisation, the information which is obtained from these competitions is still limited. Winners are frequently the teams which have devoted more time understanding the nuts and bolts of the competition and to (correspondingly) tuning their algorithms for the tasks. Also, the 'complexity' of each task is always quantified or estimated in an informal or ad-hoc way, so it is very difficult to compare results across different algorithms and competitions.

An alternative proposal for intelligence and performance evaluation is based on the notion of universal distribution [12] and the related algorithmic information theory (a.k.a. Kolmogorov complexity) [10]. Note that any universal distribution does not assign the same probability to all objects (which would be 0, since there are infinitely many), but it gives higher probability to objects with smaller descriptions. Using this theory, we can define a universal distribution of tasks for a given AI realm, and sort them according to their (objective) complexity. Some early works have developed these ideas to construct intelligence tests. First, [1] suggested the introduction of inductive inference problems in a somehow *induction-enhanced* or *compression-enhanced* Turing Test [15]. Second, [3] derived intelligence tests (C-tests) as sets of sequence prediction problems which were generated by a universal distribution, and the result (the intelligence of the agent) was a sum of performances for a range of problems of increasing complexity. The complexity of each sequence was derived from its Kolmogorov complexity (a Levin variant was used). This kind of problem (discrete sequence prediction), although typical in IQ tests, is a narrow AI realm. In fact, [11] showed that relatively simple algorithms could score well at IQ tests (and, as a consequence, at C-tests). In [3] the suggestion of using interactive tasks where "rewards and penalties could be used instead" was made. Later, Legg and Hutter (e.g. [7],[8]) gave a precise definition to the term "Universal Intelligence", as a sum (or weighted average) of performances in all the possible environments. Environments are understood as is custom in reinforcement learning. However, in order to make the extension from (static) sequences to (dynamic) environments, several issues had to be solved first. In [6], the problem of finding a finite sample of environments and sessions is addressed, as well as approximations to Kolmogorov complexity, the inclusion of time, and the proper aggregation of rewards. The theory, however, has not been applied in the form of a real test, to evaluate artificial and biological agents. This is the goal of our paper.

Since these recent approaches are constructed over a reinforcement learning (RL) setting, it seems natural to start evaluating RL algorithms. In fact, RL [14][20] is a proper and general setting to define and analyse learning agents which interact with an environment through the use of observations, actions and rewards. Hence, RL is not strictly limited to AI agents; non-human animals and humans can be understood in this setting, most especially in the context of evaluation. When trying to pick up a 'representative' algorithm to start with, we face a difficult issue, since there is a vast amount of literature on RL algorithms. According to [20], the three most influential algorithms are Temporal Difference (TD) Learning , adaptive Actor-Critics and Q-learning [17]. Here we choose Q-learning and we evaluate it in terms of the theory given in [6] and an environment class defined in [4]. We present here a first implementation of the tests and we evaluate Q-learning using these tests. The use of a general intelligence test for Q-learning provides some interesting insights into how RL algorithms could be evaluated (with a general intelligence test) and also into the viability of the test as a general intelligence test for AI.

The paper is organised as follows. Section 2 briefly describes the theory presented in [6] and the environment class introduced in [4]. Section 3 gives some details on the implementation of the tests, and introduces the types of agents we will evaluate with the test. The next sections perform an experimental evaluation, using a simple example first (section 4), showing the basic experimental results and their relation to complexity (section 5). Section 6 follows with a discussion of the results and related work, and section 7 closes the paper.

2 An Environment Class for a Universal Intelligence Test

Effective testing and evaluation of an individual's ability requires an accurate choice of items in such a way that the tests are discriminative and quantify the capability to be measured. Measuring (machine) intelligence is not different. [6] presents the first general and feasible setting to construct an intelligence test which claims to be valid for both artificial intelligent systems and biological systems, of any intelligence degree and of any speed. The test is not anthropomorphic, is gradual, is anytime and is exclusively based on computational notions, such as Kolmogorov complexity. And it is also meaningful, since it averages the capability of succeeding in different environments. The notion of environment is similar to the general notion which is used in reinforcement learning - by using actions, rewards and observations. The key idea is to order all the possible environments by their Kolmogorov complexity and use this ordering to make samples and construct adaptive tests that can be used to evaluate the intelligence of any kind of agent. The test configures a new paradigm for intelligence measurement which dramatically differs from the current task-oriented and ad-hoc measurement used both in artificial intelligence and psychometrics.

One of the key issues in the previous test is the use of discriminative environments only. That means that environments which may lead to dead-ends, are too slow, or that only allow a few interactions with the agent are ruled out. Additionally, a selection of the remaining environments must be done according to a sample of all the (infinitely many) possible environments. The choice of an unbiased probability distribution to make the sample is then crucial.

As a consequence, the choice of a proper environment class is a crucial issue. The more general the environment class, the better. This is what [4] attempts, a hopefully unbiased environment class (called Λ) with spaces and agents with universal descriptive (Turing-complete) power. Basically, the environment class Λ considers a space as a graph of cells (nodes) and actions (vertices). Objects and agents can be introduced using Turing-complete languages to generate their movements. The environment class can be summarised as follows:

- Space (Cells and Actions): The space is defined as a directed labelled graph, where each node represents a cell, and arrows represent actions. The topology of the space can vary, since it is defined by a randomly-generated set of rules (using a geometric distribution with p = 1/2). The graph is selected to be strongly connected (all cells are reachable from any other cell).
- Agents: Cells can contain agents. Agents can act deterministically (or not) and can be reactive to other agents. Agents perform one action at each interaction of the environment. Every environment must include at least three agents: the evaluated agent, and two special agents *Good* and *Evil*.
- Observations and Actions: Actions allow the evaluated agent (and other agents) to move in the space. Observations show the cell contents.
- Rewards: rewards are generated by means of the two special agents *Good* and *Evil*, which leave rewards in the cells they visit. Rewards are rational numbers in the interval $[-1, 1]$. *Good* and *Evil* have the same pattern for behaviour except for the sign of the reward (+ for *Good*, − for *Evil*). This makes *Good* and *Evil* symmetric, which ensures that the environment is balanced (random agents score 0 on average) [6]. *Good* and *Evil* are initially placed randomly in different cells (and they cannot share a cell).

For the space (the graph) and also for the behaviour of all the agents, a Turing-complete language based on rewriting rules (Markov algorithms) is proposed.

The environment class Λ is shown in [4] to have two relevant properties for a performance test: (1) their environments are always balanced, and (2) their environments are reward-sensitive (there is no sequence of actions such that the agent can be stuck in a heaven or hell situation, where rewards are independent of what the agent may do). As argued in [6], these two properties are very important for the environments to be discriminative and comparable (and hence the results being properly aggregated into a single performance or intelligence score). No other properties are imposed, such as (e.g.) environments being Markov processes or being ergodic.

Several interfaces have been defined so that we can test biological subjects and machines. In this paper we will focus on evaluating machine algorithms. For more details of the environment class Λ, see [4].

3 Implementation and Evaluated Agents

Following the definition of the environment class Λ, we generate each environment as follows. Spaces are generated by first determining the number of cells n_c, which is given by a number between 2 and 9, using a geometric distribution (i.e. $prob(n) = 2^{-n}$, and normalising to sum up to 1). Similarly, the number of actions n_a is defined with a geometric distribution between 2 and n_c. Both cells and actions are indexed with natural numbers. There is a special action 0 which connects every cell with itself (i.e., to stay at the cell). The connections between cells are determined using a uniform distribution for each cell, among the possible actions and cells. We consider the possibility that some actions do not lead to any cell. These actions have no effect. A cell which is accessible from another cell (using one action) is called a 'neighbouring' or adjacent cell. An example of a randomly generated space can be shown in Fig. 1.

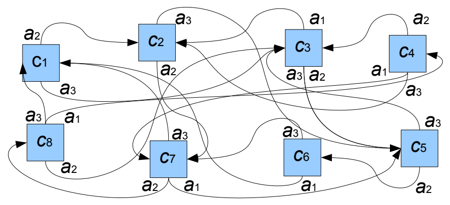

Fig. 1. A space with 8 cells and 4 actions (a_0, a_1, a_2, a_3). Reflexive action a_0 not shown.

The number of cells and actions is, of course, related to the complexity of the space, but not monotonically related to its Kolmogorov complexity (or a computable variant such as Levin's Kt [9]). Nonetheless, most of the actual grading of environments comes from the behaviour of *Good* and *Evil*. The sequence of actions for *Good* and *Evil* is defined by using a uniform distribution for each element in the sequence, and a geometric distribution (p = 1/100) to determine whether to stop the sequence, by using a probability of stopping (p_{stop}). An example of a sequence for the space in Fig. 1 is 203210200, which means the execution of actions a_2, a_0, a_3, a_2, etc. If *Good* is placed at cell c_5, the pattern will lead it (via a_2) to c_6 in the next step, since it starts with '2'. The agents *Good* and *Evil* take one action from the sequence and execute it for each step in the system. When the actions are exhausted, the sequence is started all over again. If an action is not allowed at a particular cell, the agent does not move.

Given a single environment, evaluation is performed in the following way. Initially, each agent is randomly (using a uniform distribution) placed in a cell. Then, we let *Good*, *Evil*, the evaluated agent and any other agents in the space interact for a certain number of steps. We call this a session. For a session we average the rewards, so giving a score of the agent in the environment.

Although [4] suggests a partially-observable interface, here we will make it fully-observable, i.e., the agents will be able to see all the cells, actions and contents. Rewards are not part of the observation and hence are not shown.

And now we present the agents we will evaluate.

– Random: a random agent is just an agent which chooses randomly among the available actions using a uniform distribution.
– Trivial Follower: this is an agent which looks at the neighbouring cells to see whether *Good* is in one of them. If it finds it, then it moves to that cell. Otherwise, trying to avoid *Evil* it makes a random move.
– Oracle: this agent 'foresees' the cell where *Good* will be at the next step and if this cell is one of the neighbouring cells then it moves to that cell. Otherwise, it goes to the adjacent cell that, in the next iteration, will have the highest reward. Even though the 'oracle' has a sneaky advantage over the rest of the agents, it is not an 'optimal' agent, since it only foresees one-step movements, and this may be a bad policy occasionally.
– Q-learning: this is an off-the-shelf implementation of Q-learning, as explained in [17] and [14]. We use the description of cell contents as a state. Q-learning has two classical parameters: *learning rate* α and *discount factor* γ.

The choice of Q-learning as an example of a reinforcement learning algorithm is, of course, one of many possible choices, from a range of other RL algorithms from the same or different families. The reason is deliberate because we want a standard algorithm to be evaluated first, and, most especially, because we do not want to evaluate (at the moment) very specialised algorithms for ergodic environments or algorithms with better computational properties (e.g. delayed Q-learning [13] would be a better option if speed were an issue).

The parameters for Q-learning are $\alpha = 0.05$, $\gamma = 0.35$. The elements in the Q matrix are set to 2.0 initially (rewards range from -1 to 1, but they are normalised between 0 to 2 to always be positive in the Q matrix). The parameters have been chosen for the set of experiments in this paper by trying 20 consecutive values for α and γ between 0 and 1. These 20 x 20 = 400 combinations have been evaluated for 1,000 sessions each using random environments.So the parameters have been chosen to be optimal for the set of experiments included in this paper.

4 A Simple Example

Given the description of how environments are generated we will show how the previous agents perform in a single environment. The environment is based on the space in Fig. 1 and the following sequence for *Good* and *Evil*: 203210200. The number of steps (iterations) has been set to 10,000.

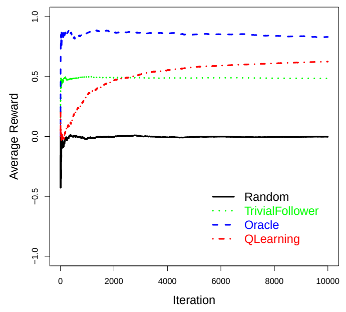

Fig. 2. Results for a simple environment over the 8-cell space in Fig. 1 and the sequences of actions that *Good* and *Evil* will follow: 203210200

The results of average reward for the several types of agents seen in fig. 2 show that initially the average reward has great fluctuations for the four types of agents. The random agent soon converges to its expected average reward, which is 0. The trivial follower is only able to score close to 0.5. In this environment and pattern, following *Good* is a good policy, but only to some extent. The oracle converges to a value around 0.83. As mentioned above, the oracle is near-optimal, and in many environments it will not reach a value of 1, but just a good value. Q-learning results in a slow convergence to a value of around 0.625. Although slow, in this case we see that it outperforms the trivial follower in the end.

Nonetheless, these results are only given for one particular environment. The following sections perform a battery of experiments which try to obtain some conclusions about the general behaviour of these agents.

5 Experiments

In this section we maintain the parameters and settings described in previous sections but now we average the results over many environments. In particular, we choose 100 environments of 3 cells and 100 environments of 9 cells, which allow us to summarise the results for a range of cells between 3 and 9. Each environment has a random generation of the topology and a random generation of the sequence for *Good* and *Evil* as described above. The probability of stopping p_{stop}, which controls the size of the pattern, is set to 1/100.

The results are shown in Fig. 3. From these figures we get a general picture which is consistent with the single example shown above. Now we do not see fluctuations, since these are average results for 100 experiments each. We see that once contact is made with *Good*, then the policy is to follow it (somehow

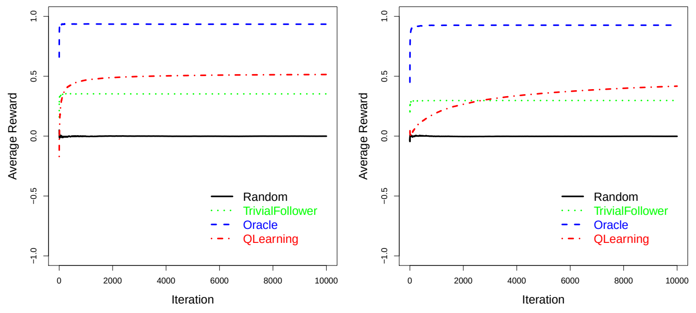

Fig. 3. Results for 100 environments. Left: 3 cells. Right: 9 cells.

unsuccessfully for the trivial follower and very successfully for the 'tricky' oracle). Q-learning is able to surpass the trivial follower (by taking advantage of the pattern for *Good* and *Evil*) but just after a slow convergence, which is slower the higher the number of cells is, as expected.

We analyse the effect of complexity. In Fig. 4, we show the reward results of the four algorithms after 10,000 iterations compared to the 'complexity' of the environment. In order to approximate this complexity, we use the size of the compression of the description of the space (which obviously depends on the number of cells and actions), denoted by S, and the description of the pattern for *Good* and *Evil*, denoted by P. More formally, given an environment μ, we approximate its (Kolmogorov) complexity, denoted by K^{approx}, as follows:

$$K^{approx} = LZ(concat(S, P)) \times |P|$$

For instance, if the 8-cell 4-action space seen in Fig. 1 is coded by the following string $S =$ "`12+3----- | 12+++++3----- | 1-2------3++ | 1-----2++++++3- | 12+3++++++ | 1-----23------- | 1++++++2-------3++ | 1----2+++3+`" and the pattern for *Good* and *Evil* is described by $P =$ "`203210200`", we concatenate both strings (total length 119) and compress the string (we use the 'gzip' method given by the *memCompress* function in R, a GNU project implementation of Lempel-Ziv coding). The length of the compressed string in this case is 60.

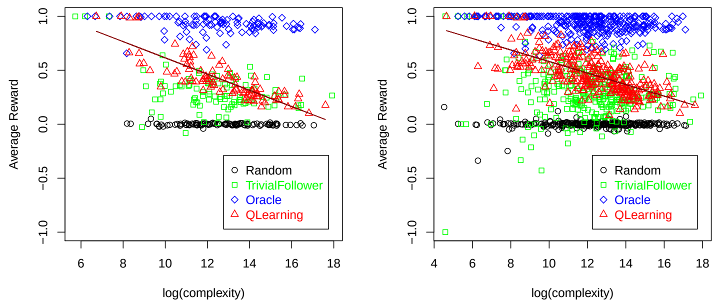

Fig. 4. Results after 10,000 iterations compared to the complexity of the environment. Left: 100 environments with 9 cells. Right: 300 environments with (3, 6, 9) of cells.

We see that the random agent, the oracle and the trivial follower are barely affected by complexity. On the contrary, Q-learning shows a significant decrease in performance as long as complexity is increased.

6 Discussion and Related Work

The experiments shown in previous sections portray a view about how an implementation of the intelligence test introduced in [6] using the environment class Λ presented in [4] can be used to evaluate AI systems. Although the implementation has used several approximations and simplifications, we show that the theory works in capturing the essence of task complexity independently of the particular AI application or AI system to be evaluated.

Naturally, the application to the evaluation of RL systems is much more straightforward than other AI systems, since the test framework and reinforcement learning are based on the notion of interacting with an environment through observations, actions and rewards. Nonetheless, we think that most (if not all) tasks and areas in AI can be re-framed using this framework. In fact, in the previous tests, we have not made any single decision to favour or to specialise for any kind of learning algorithm or agent technology. Environment complexity is based on an approximation of Kolmogorov complexity, and not on an arbitrary set of tasks or problems. Consequently, our notion of complexity is not based on the idea of aliasing, Markov property, number of states, dimension, etc. Of course, all these issues are related to the notion of complexity, but we do not restrict to any subset of problems, or any notion of convergence, computational class, etc. Originally, the test just aims at using a Turing-complete environment generator, and it is the grading given by its (Kolmogorov) complexity which allows us to aggregate the performance obtained in each environment.

It is important to compare this approach to the traditional approach in artificial intelligence. There are some works on the evaluation of problem complexity and also many AI competitions. For instance, Zatuchna and Bagnall [21] analyse mazes used in research in the last two decades, develop a specific measure of "complexity", and try to determine which kind of learning agents behave best depending on the complexity, by also using a specific measure of performance (based on correctness, convergence, and memory). Extensive works, such as Zatuchna and Bagnall's paper, are not frequent (because they require an enormous amount of work), but they are crucial for the real evaluation of progress in artificial intelligence. In our opinion, these evaluation works would be much more productive if they could be standardised under a grounded and common measurement of performance using the theory presented in [6]. In fact, what we have done here for reinforcement learning could also be restricted to 'mazes', using a proper environment class only representing mazes. Other areas such as multi-agent environments [18] could be adapted as well.

AI competitions are also a typical approach to evaluating progress in AI, such as, e.g., the AAAI General Game Playing Competition [2], or the RL-competition [19]. The latter is the closest approach to what we have done here. The RL-competition consists of several tasks for several domains. It is, in fact, several RL-competitions, one for each domain. Some environments are very specific. Others are a little bit more general, such as 'polyathlon', a set of 'normalised' classic and new reinforcement learning problems that look identical in terms of their task specification, so that the agent is not able to 'identify' which task it faces. This bottom-up approach is valuable, but we think that our top-down (theory-derived) approach is much more general and able to evaluate any kind of RL (or AI) agent without the risk of having systems specialised to it.

7 Conclusions

The goal of the paper was not to analyse some well-known properties of Q-learning (such as convergence, state overloading, etc.) – nor to designate a 'winning' algorithm. The goal of the paper, rather, was to show that a top-down (theory-derived) approach for evaluating AI agents can work in practice. We have used an implementation of [6] to evaluate Q-learning, as a typical off-the-shelf algorithm. We have seen the (inverse) relation of Q-learning performance with environment complexity. No restrictions about aliasing problems, partial observability, number of states, Markov properties, etc., are made here. As a direct application, several AI competitions and evaluation repositories could be defined using appropriate environment classes. The evolution of different algorithms with respect to the environment complexity would be one key feature to examine and a better indicator of progress in AI.

There is, of course, much work ahead. One clear area for future work is the evaluation of other reinforcement learning algorithms and the analysis of the parameters in all these algorithms (including Q-learning). In order to do this, we plan to integrate our system into the RL-glue architecture, so we could easily apply our tests to many existing RL algorithms already implemented for the RL-glue platform. One algorithm we want to evaluate soon is a Monte Carlo approximation to AIXI [16], which is showing impressive learning performance on complex tasks and, interestingly, is based on ideas derived from Solomonoff prediction and Kolmogorov complexity. Another line for future work is to progress on a new version of the implementation of the test which could be more adherent to its full specification, by using better Turing-complete environment generators and better approximations for complexity. In this context, the implementation of the anytime version of the test in [6] (using the aggregation introduced in [5]) would also allow us to compare algorithms using efficiency as an important factor of the performance of the algorithms.

Finally, using our tests for humans and (non-human) animals will also be a very important source of information to see whether this top-down approach for measuring performance and intelligence can become mainstream in AI.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We also thank José Antonio Martín H. for helping us with several issues about the RL competition, RL-Glue and reinforcement learning in general. We are grateful for the funding from the Spanish MEC and MICINN for projects TIN2009-06078-E/TIN, Consolider-Ingenio CSD2007-00022 and TIN2010-21062-C02, for MEC FPU grant AP2006-02323, and Generalitat Valenciana for Prometeo/2008/051.

References

1. D. L. Dowe and A. R. Hajek. A non-behavioural, computational extension to the Turing Test. In *Intl. Conf. on Computational Intelligence & multimedia applications (ICCIMA'98), Gippsland, Australia*, pages 101–106, 1998.
2. M. Genesereth, N. Love, and B. Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2):62, 2005.
3. J. Hernández-Orallo. Beyond the Turing Test. *J. Logic, Language & Information*, 9(4):447–466, 2000.
4. J. Hernández-Orallo. A (hopefully) non-biased universal environment class for measuring intelligence of biological and artificial systems. In M. Hutter et al., editor, *Artificial General Intelligence, 3rd Intl Conf*, pages 182–183. Atlantis, 2010.
5. J. Hernández-Orallo. On evaluating agent performance in a fixed period of time. In M. Hutter et al., editor, *Artificial General Intelligence, 3rd Intl Conf*, pages 25–30. Atlantis Press, 2010.
6. J. Hernández-Orallo and D. L. Dowe. Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence*, 174(18):1508 – 1539, 2010.
7. S. Legg and M. Hutter. A universal measure of intelligence for artificial agents. In *Intl Joint Conf on Artificial Intelligence, IJCAI*, volume 19, page 1509, 2005.
8. S. Legg and M. Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17(4):391–444, 2007.
9. L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.
10. M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications (3rd ed.)*. Springer-Verlag New York, Inc., 2008.
11. P. Sanghi and D. L. Dowe. A computer program capable of passing IQ tests. In *Proc. 4th ICCS International Conference on Cognitive Science (ICCS'03), Sydney, Australia*, pages 570–575, 2003.
12. R. J. Solomonoff. A formal theory of inductive inference. Part I. *Information and control*, 7(1):1–22, 1964.
13. A.L. Strehl, L. Li, E. Wiewiora, J. Langford, and M.L. Littman. PAC model-free reinforcement learning. In *Proc. of the 23rd Intl Conf on Machine learning*, ICML '06, pages 881–888, New York, 2006.
14. R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* The MIT press, 1998.
15. A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.
16. J. Veness, K.S. Ng, M. Hutter, and D. Silver. Reinforcement learning via AIXI approximation. In *Proc. 24th Conf. on Artificial Intelligence (AAAI-10)*, pages 605–611, 2010.
17. C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.
18. D. Weyns, H.V.D. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for multi-agent systems, state-of-the-art and research challenges. In *Environments for MAS*, volume 3374 of *LNCS*, pages 1–48. Springer, 2005.
19. S. Whiteson, B. Tanner, and A. White. The Reinforcement Learning Competitions. *The AI magazine*, 31(2):81–94, 2010.
20. F. Woergoetter and B. Porr. Reinforcement learning. *Scholarpedia*, 3(3):1448, 2008.
21. Z. Zatuchna and A. Bagnall. Learning mazes with aliasing states: An LCS algorithm with associative perception. *Adaptive Behavior*, 17(1):28–57, 2009.