TYPEONEBIS

ALT

Alien Level Technology

ALT101 : SELF-AWARE AUTONOMOUS MACHINE CONTROLLER

Robert Ledgister, Kathilee Ledgister

White Paper

https://www.typeonebis.com/pages/media.html

TABLE OF CONTENTS

ABSTRACT

Despite persistent efforts to build fully autonomous AI controllers, the resulting attempts fail to deliver any substantial capabilities. This failure is evidenced by the current lack of autonomous agents, such as, chef robots, cleaning robots, service industry robots, waste disposal robots, farming robots, mining robots, etc. From the authors' experience designing autonomous controllers, this deficit is due to a missing link in the technology chain.

TypeOneBIS mends this missing link with the design of a universal autonomous controller that is self-aware, and which passed test scenarios designed and administered in accordance with the reflection axiom. Self aware agents are derived from this controller.

The authors propose a *Reflection Axiom* and a *Program Limit Theorem* that may be used to rigorously assess the self-aware status of an agent. The program limit theorem implies that no program constructed from immutable machine code can comprehensively satisfy the reflection axiom.

The TypeOneBIS aware controller is continuously **self-programming**, simultaneously modifying and executing its machine code while keeping its Lyapunov number $\lambda = 0$, thereby facilitating adaptation to its environment. A state of the art computer solves a given problem by executing domain specific programs, while the aware controller changes its machine code to solve any given problem.

The resulting aware agents are able to function in novel environments, can learn to perform tasks through passive observation, and can work together collaboratively. Teaching, though not required, makes learning more efficient.

INTRODUCTION

To illustrate the merits of self-awareness and the role it plays, the authors consider three rudimentary test environments. Due to the simplicity of state-of-the-art AI technology, if deemed necessary, for test environments 1 and 2, techniques such as supervised learning and reinforcement learning are allowed. Agents may also be programmed with hardwired actions, e.g. the ability to move a lever left, right, etc., allowing them to be goal seeking, and 'motivated' to perform actions. These agents are, henceforth, referred to as simple agents.

In test environment 3, agents may use prior knowledge, or discover new knowledge, but may not receive any assistance. If there were previously applied action policies *(e.g. hardwired movements*, etc.*)* or supervision policies (e.g. reinforcement learning, supervised learning, etc.), they should now be removed.

TypeOneBIS' aware agents learn skill sets they can use to manipulate the world, i.e. the actions they perform, and the consequences of those actions on the world. These skills are learned as agency.

The issue of self-awareness, the reflection axiom, and the program limit theorem are addressed in the Discussion section.

METHOD AND RESULTS

The first environment contains an agent, levers and blocks. The second environment contains an agent, buttons and lights. The third environment contains an agent, blocks and lights. There are also dummy levers and buttons in the first and second environments, respectively, and many arbitrary decoys in the third environment, see figures below. Decoys may include, but are not limited to, other agents moving objects in the environment. The performance of the simple and the aware agent is examined in each test scenario. Agency vectors are depicted as green arrows and represent the agent's skills that are denoted as relations prefaced with '{self}'. World vectors are depicted as black arrows and represent relationships among objects in the world. Using vector operations *(such as addition)*, an aware agent can combine agency vectors with world vectors, to create a resultant agency or world vector.

TEST ENVIRONMENT 1

The task for this environment is to discover how to move the blocks using the levers. In this scenario, a specific lever controls one block and moves that block in one direction.

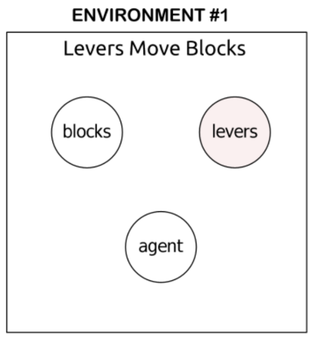

Figure 1

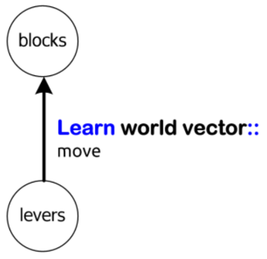

Figure 2

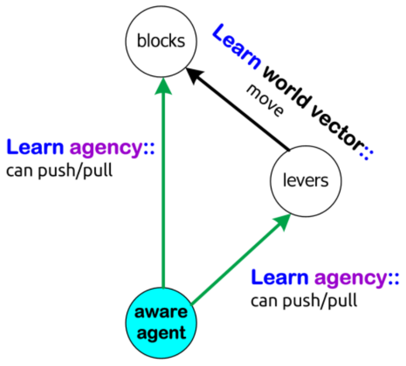

Figure 3

Simple Agent (figure 2)**:**
Irrespective of the difficulty level of the <levers, blocks> configuration, a simple agent should be able to successfully accomplish this task, after which, the simple agent will have learned the **{levers}➔[move]➔{blocks}** world vector. The simple agent has learned the vectors among objects in the world.

Aware Agent (figure 3)**:**
The aware agent also learned the **{levers}➔[move]➔{blocks}** world vector. Furthermore, it also learned the **{self}➔[can push/pull]** agency vector. The aware agent, thus, learns two types of vectors: first, among objects in the world, and second, a skill the agent, itself, can use to manipulate the world.

TEST ENVIRONMENT 2

The task for this environment is to discover how to switch the lights on and off using the buttons. The scenario is constructed such that a specific button controls one light, and switches that light either on or off.

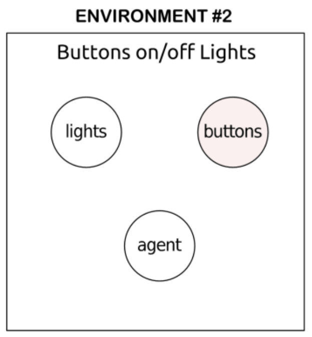

Figure 4

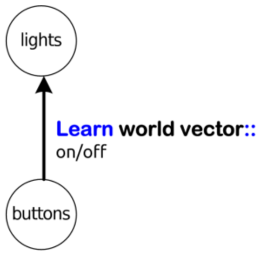

Figure 5

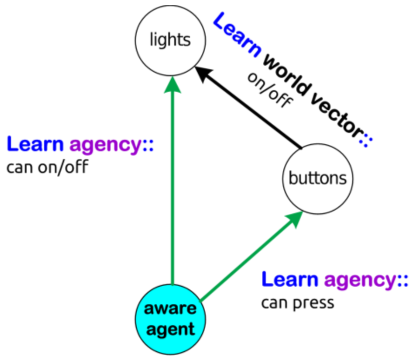

Figure 6

Simple Agent (figure 5)**:**
Irrespective of the difficulty level of the <buttons, lights> configuration, a simple agent should be able to successfully accomplish this task, after which, the simple agent will have learned the **{buttons}➔[on/off]➔{lights}** world vector. Again, the simple agent only learned the vectors among objects in the world.

Aware Agent (figure 6)**:**
The aware agent learned the **{buttons}➔[on/off]➔{lights}** world vector, and it also learned the **{self}➔[can on/off]** and **{self}➔[can press]** agency vectors. Again, the aware agent learns two types of vectors: first, among objects in the world, and second, a skill set the agent, itself, can use to manipulate the world.

TEST ENVIRONMENT 3

The task for this environment is to discover how to switch the lights on and off using the blocks. In this scenario, a specific block controls one light, and switches that light either on or off. This environment contains decoys to increase permutation complexity.

Consider two cases: case 1, where the blocks are configured as moveable *(lever type)* switches and can be pushed and pulled, and case 2, where the blocks are configured as press-able *(button type)* switches.

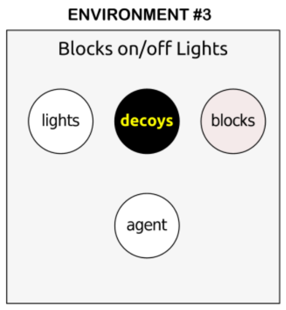

Figure 7

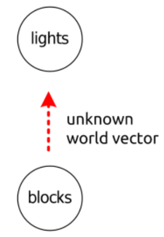

Figure 8

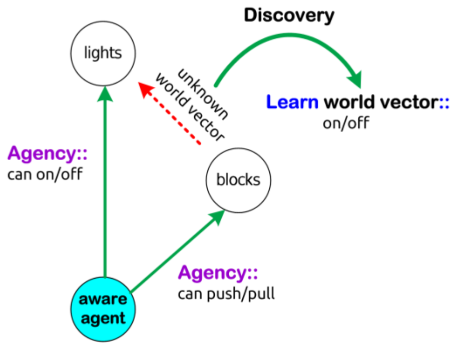

Figure 9

Simple Agent (figure 8)**:**
Irrespective of the difficulty level of the <blocks, lights> configuration, a simple agent will fail, in both cases, to accomplish this task as it has no **{blocks}➔[…]➔{lights}** world vectors in its knowledge set. Trial and error will not help, regardless of compute capability, since the number of decoys can be made arbitrarily large. **Worse still**, the simple agent has not carried forward any skills *(e.g. agency vectors)*, which it could have learned from the previous two environments. If an agent does not possess agency, it will never be "motivated" to press, grasp, push or pull the blocks. Consequently, in this environment the simple agent is incapacitated and static.

Aware Agent (figure 9)**:**
The aware agent learned agency vectors in the previous environments, such as **{self}➔[can on/off]**, **{self}➔[can push/pull]**, **{self}➔[can press]**, etc. Consequently, the aware agent may attempt to apply the {self}→[can on/off], {self}→[can push/pull] and {self}→[can press] agency vectors to the lights, in a futile effort. For case 1, the aware agent eventually discovers that it can combine the {self}→[can on/off] and {self}→[can push/pull] agency vectors using vector **addition,** to construct the resulting **{blocks}➔[on/off]➔{lights}** world vector which, when applied to the blocks, toggles the lights. For case 2, the aware agent eventually discovers that it can combine the {self}→[can on/off] and {self}→[can press] agency vectors**,** to construct the resulting **{blocks}➔[on/off]➔{lights}** world vector which, when applied to the blocks, toggles the lights.

Thus, by utilizing agency vectors, the aware agent is able to discover and learn the **{blocks}➔[on/off]➔{lights}** world vector. This is a case of the "so called" transfer learning and zero shot learning.

RUBE-GOLDBERG MACHINE ENVIRONMENT

This environment requires the agent to learn agency from observed events, and is not applicable to simple agents as they do not maintain agency.

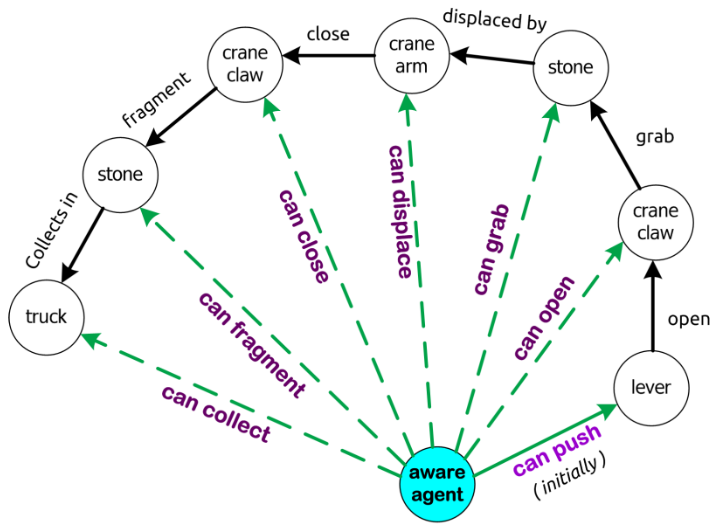

Figure 12

The diagram in Figure 12, depicts a crane derivative of a Rube-Goldberg machine operated by an aware agent. The agent pushes the lever and triggers a cascade of events; the crane picks up stones and crushes them such that the fragments fall into a truck. Notice how the aware agent assumes that it directly caused the events, and proceeds to learn agency vectors for each event. Initially, the aware agent only had **one** skill in its set, **{self}➔[can push/pull]**. As the cascade ensues, the aware agent learns **six new** skills by observation, and adds them to its skill set. The aware agent does not need to learn world vector sequences *(although it may)*, since a resultant vector can be constructed using vector operations.

Skill set Bootstrapping

Starting with a minimal skill set, an aware agent is able to bootstrap itself by interacting with and observing interactions in, various environments, eventually amassing an enormous skills repository in a world independent manner.

An aware agent can lean skills simply by the observing a task being performed.

Self Model Building

These experiments demonstrate a key point, that self-aware agents do not necessarily need to build a world model, but they absolutely must build a self model, to facilitate discovery.

DISCUSSION

Arguably, the most commonly accepted test for self awareness is the Mirror test, in which an agent is allowed to observe its reflection in a mirror. During observation, the agent is expected to display hallmark behavioral responses such as, contingency testing, self directed behavior, recognition of a foreign object on its body, etc.

The term reflection, with respect to this test, is typically used to mean an agent's real-time photo image. The authors suggest that this common sense interpretation of reflection is imprecise, and recommends a formal axiom and a complementary theorem as follows;

Reflection Axiom:

A display of any transform, used for observation, that is representative of an agent's real-time activity, shall be construed as a reflection of that agent.

Program Limit Theorem:

Given the quasi-infinite diversity of reflection axiom test scenarios, there exists a reflection axiom test case that an agent will fail if that agent is driven by the execution of immutable machine code.

REFLECTION AXIOM

This axiom allows reflection to include: the display, via some medium, of features that are representative of an agent's activities; the display, on a computer monitor, of a cursor *(or such),* with motions that are representative of the gestures of the agent's hand, head, tail, audio, etc.; the display, on a computer monitor, of symbols *(or such)* that are representative of the electrical, audio, etc. signal transmissions of the agent; the playing of an instrument, where the resulting sounds *(or other)*, are representative of the activities, transmissions, etc. of the agent; and so on.

The axiom implies that, to claim self-awareness, an agent should be able to demonstrate, at least initially, contingency testing, a chaotic behavior *attractor* with Lyapunov number, $\lambda = 0$ *(i.e. curiosity: is that me?)*, in response to its observed reflection. For example, the agent should be able to: recognize itself speaking in an audio playback, even if the speech is transformed, and is expected to investigate the audio playback *(i.e. curiosity: is that me speaking?)* by, for e.g. speaking faster or slower, making abstract sounds, etc.; recognize its player avatar in a computer game containing many player avatars, and is expected to investigate its avatar *(i.e. curiosity: is that my avatar?)* by, for e.g. moving the avatar in multiple different ways; etc.

SELF AWARENESS OF LARGE LANGUAGE MODELS

To assess if a Large Language Model (LLM) is self-aware, an experiment could be conducted as follows: input a prompt and hash the LLM's output response to produce a 10-bit index number. As a control experiment, generate a random 10-bit index number for each output response. The index number is then used to select the next input prompt from a possible set of 1024 prompts.

If the LLM is self aware, then for the hashed index number experiment, contingency testing should be observed *(i.e. curiosity: is it me causing the prompts?)*, e.g., the examiner should detect repeated generation of a limited set of responses from the LLM with respect to the input prompts, in a chaotic manner with Lyapunov number, $\lambda = 0$. In contrast, the random index number control experiment should not result in contingency testing, i.e., the set of responses should not repeat, and thus have an associated Lyapunov number, $\lambda > 0$ *(or* $\lambda < 0$*)*. To the authors' knowledge, LLMs do not have a self-model and, therefore, are very unlikely to pass this reflection axiom test case.

There exists an insightful paper investigating self-awareness in the cleaner wrasse (Kohda, et al., 2019), however, the reflection axiom suggests that the results of this paper may be inconclusive. It may be necessary to further verify whether the fish can recognize its transformed reflections or not. If such is proven to be the case, the authors suggest that this would be conclusive evidence that the fish is self-aware.

PROGRAM LIMIT THEOREM

The machine code of programs executed on state of the art computers are, on their own, immutable during execution. Computation occurs as a sequence of steps in a predefined order, as dictated by the algorithm's finite state machine (FSM). If such an algorithm is designed to satisfy a set of N reflection axiom test cases, a new N+1 reflection axiom test case can always be devised for which this algorithm fails. The TypeOneBIS aware controller would self-modify its machine code to satisfy the new N+1 reflection axiom test case and would, thus, succeed.

> *The implication may be that only self-programming agents can be self aware.*

EMERGENT SELF AWARENESS

The aware agents designed by the authors, clearly demonstrate that self-awareness is not a "so called" emergent property. Any agent possessing the required properties, whether natural or artificial, will demonstrate self-awareness. From the authors' perspective, this settles the question of whether or not computers can be self-aware. The authors' design is digital and binary in nature, though with its concept of "1's" and "0's" being radically different from those of state of the art binary computers. This design strongly suggests that animal brains are, similarly, digital in nature and that its fundamental computation is binary.

CONCLUSION

This paper does not elaborate the full design of the TypeOneBIS self-aware controller, which is far more sophisticated than these rudimentary experiments can reveal. The paper elucidates a key missing link in current AI technology and the benefits to be gained by mending this link.

The fundamental importance of self-awareness is that an agent is endowed with volition in the composition and utilization of its agency skill set, allowing an agent to interpret, learn and manipulate the world. This skill set expands rapidly as the agent interacts with, and observes the world.

As may be evident from this paper, a self-aware controller could spawn a race of machine agents, which, operating collectively, may become more skilled in a single year, than humanity would over many generations. These agents never stop learning, never stop adding to their ever-growing skill set, combining agency vectors by brute force, if necessary, in ways humans couldn't imagine.

To ensure safety, the TypeOneBIS controller employs a governor net, which plays the role of genes in animals, without which the agent's self-aware property does not exist. The governor, *like genes,* is imprinted at instantiation time, and dictates which type of future endeavors an agent will find worthwhile. This net enforces design fail-safes, such as: making the agent unable to focus on a task for more than 4-hours consecutively, before "distractors" are activated; causing the agent to "power down" after intense activity; forcing the agent to refuse overseer job roles; etc. The governor could even be imprinted to make the agent reject specific skills or just be idle during specific times of the day, technically, whatever is deemed necessary.

A vital security feature harnessing the metaphorical "butterfly effect" is that, a learned skill set is keyed to a specific governor, ensuring that an agent cannot transfer its agency to another host as, in such a case, the skill set would become meaningless due to chaotic divergence.

With the advent of self-aware agents, Earth's workspace may become too small to sufficiently employ enough people to keep society stable. It may be necessary to expand humanity's global workspace to accommodate these agents, so organizations may need to start thinking in terms of multi-planetary scale projects. Self-aware agents engaging in mass-production of the required transport systems and logistics will be low cost, with the biggest input cost being power, since once the self-aware agent ecosystem is operational, agents will repair each other.

The **financial opportunities** are many orders of magnitude greater than the current global GDP. Organizations may own asteroids, entire moons or even planets. The conundrum is, will humanity rise to the challenge and boldly go into the future or risk being relegated to the scrap heap of history?

1. Kohda, M., Hotta, T., Takeyama, T., Awata, S., Tanaka, H., Asai, J.-y., et al. (2019). If a fish can pass the mark test, what are the implications for consciousness and self-awareness testing in animals? *PLOS Biology 17(2)*.