EGPAI 2016

1st International Workshop on
Evaluating General-Purpose AI

The Hague, The Netherlands,
Tuesday 30th August 2016

A workshop held in conjunction with ECAI 2016

Accepted Papers

More information at:
`http://users.dsic.upv.es/~flip/EGPAI2016/`

:

A Dynamic Intelligence Test Framework for Evaluating AI Agents

Nader Chmait and **Yuan-Fang Li** and **David L. Dowe** and **David G. Green**[1]

Abstract. In our recent work on the measurement of (collective) intelligence, we used a dynamic intelligence test to measure and compare the performances of artificial agents. In this paper we give a detailed technical description of the testing framework, its design and implementation, showing how it can be used to quantitatively evaluate general purpose, single- and multi-agent artificial intelligence (AI). The source code and scripts to run experiments have been released as open-source, and instructions on how to administer the test to artificial agents have been outlined. This will allow evaluating new agent behaviours and also extending the scope of the test. Alternative testing environments are discussed along with other considerations relevant to the robustness of multi-agent performance tests. The intuition is to encourage people in the AI community to quantitatively evaluate new types of heuristics and algorithms individually and collectively using different communication and interaction protocols, and thus pave the way towards a rigorous, formal and unified testing framework for general purpose agents.

1 INTRODUCTION

One of the major research questions at the present state of artificial intelligence is: how smart groups of artificial agents are compared to individual agents? Measuring machine intelligence is a complex topic that has been tackled by a large number of theories and is not yet fully understood as discussed in [24], [22, Section 2] and [21, Chapter 5]. Besides that, the design and study of agent systems has widely expanded in the last few years as agent models are increasingly being applied in a wide range of disciplines in the purpose of modelling complex phenomena.

In our previous work on the measurement of collective intelligence [9, 7, 8] we identified a range of factors that hinder the effectiveness of individual and interactive AI agents. This was achieved by implementing a dynamic intelligence test based on the literature of artificial general intelligence and algorithmic information theory [23, 22, 25]. We have used it to measure and compare the performance of individual, and collectives of, artificial agents across several environmental and interactive settings.

In this paper we give detailed technical description of our dynamic intelligence testing framework. We discuss its design and implementation and show how it can be used to quantitatively evaluate general purpose AI individually, and also collectively using various interaction protocols. As anticipated in our recent work [9], the source code and scripts to run experiments have been released as open-source, and instructions on how to administer the test to artificial agents have been outlined. Consequently, it is now possible to evaluate new agent behaviours, and easily extend the scope of the evaluation framework. The intuition is to encourage people in the AI community to evaluate new types of heuristics and algorithms in individual and collective scenarios using different communication and interaction protocols. This will hopefully pave the way towards a rigorous, formal and unified testing framework for general purpose artificial agents.

2 BACKGROUND

Perhaps a good start to understand the history of machine intelligence would be to take a look back at the imitation game [43] proposed by Turing in the 1950s where the idea is to have one or more human judges interrogating a program through an interface, and the program is considered intelligent if it is able to fool the judges into thinking that they are interrogating a human being. While this was once regarded as an intelligence test for machines, it has limitations [34] and is mainly a test of humanness. The machine intelligence quotient (MIQ) using fuzzy integrals was presented in [1] in 2002. However, determining a universal *intelligence quotient* for ranking artificial systems is not very practical and is almost unmeasurable due to the vast non-uniformity in the performances of different types of artificial systems. Several studies [5, 11, 12, 14, 25, 36] have investigated the relevance of compression [11, Sections 2 and 4], pattern recognition, and inductive inference [13, Section 2] to intelligence. Shortly after [5, 11, 12, 25] came the C-test [19] which was one of the first attempts to design an intelligence test consisting of tasks of quantifiable complexities. However, the test was static (non-dynamic) and it did not fully embrace the vast scope implicit in the notion of intelligence. In 2007, Legg and Hutter proposed a general definition of *universal (machine) intelligence* [30], and three years later a test influenced by this new definition, namely the *Anytime Universal Intelligence Test*, was put forward by Hernandez-Orallo and Dowe [22] in order to evaluate intelligence. The test was designed to be administered to different types of cognitive systems (human, animal, artificial), and examples environment classes illustrating the features of the test were also suggested in [22].

To the best of our knowledge, further to single agent intelligence [1, 19, 30, 22], no *formal intelligence tests* were developed in the purpose of *quantifying the intelligence of groups of interactive agents* against isolated (non-interactive) agents - which is one of the motivations behind this work. Yet, before we proceed with the description of our work, one question that might come to a reader's mind is: can't we simply evaluate and compare artificial systems over any given problem or environment from the literature? There are several reasons why we can't do that, most of them were studied and examined in [22]. We briefly summarise some of these principles. Firstly, there is a risk that the choice of the environment used

[1] Faculty of IT, Clayton, Monash University, Australia, email: {nader.chmait,yuanfang.li,david.dowe,david.green}@monash.edu

for evaluation is biased, and that it favours particular types of agents while it is unsuitable for others. Furthermore, the environment should handle any level of intelligence in the sense that dull or brilliant, and slow or fast systems can all be adequately evaluated. The test should return a score after being stopped at any time-period, short or long. Besides, not every evaluation metric is a formal intelligence test or even at a minimum, a reliable performance metric. For instance, the testing environment should be non-ergodic but reward-sensitive with no sequence of actions leading to heaven (always positive) or hell (always negative) scoring situations, and balanced [20] in the sense that it must return a null reward to agents with a random behaviour, etc.

Although the principle advantage of this work is measuring the intelligence of artificial agents, the outcome also has implications for agent-based systems. This is because it provides an opportunity to predict the effectiveness (and expected performance) of existing artificial systems under different collaboration scenarios and problem complexities. In other words, it's one way of looking at (quantifying) the emergence of intelligence in multi-agent systems.

We begin by introducing our methodology for evaluating intelligence using an agent-environment architecture (Section 3). We then re-introduce and elaborate on the Λ^* (Lambda Star) testing environment structure described in [9] (Section 4). We discuss the test implementation, its setup and parameters, in Section 5 and also give examples of how to define and evaluate new agent behaviours over the proposed testing environment. We then discuss in Section 6 some alternative testing environments that might be useful to quantify the performance of artificial agents and raise some arguments and considerations relevant to the robustness of multi-agent performance tests. We conclude in Section 7 with a brief summary.

3 AGENT-ENVIRONMENT FRAMEWORK

A common setting in most approaches to measuring intelligence is to evaluate a subject over a series of problems of different complexities and return a quantitative measure or score reflecting the subject's performance over these problems [22]. In artificial systems, the agent-environment framework [30] is an appropriate representation for this matter. For instance, this framework allows us to model and abstract any type of interactions between agents and environments. It also embraces the *embodiment thesis* [2] by embedding the agents in a flow of observations and events generated by the environment.

Here we define an environment to mean the world where an agent π, or a group of agents $\{\pi_1, \pi_2, \ldots, \pi_n\}$, can interact using a set of observations, actions and rewards [30]. The environment generates observations from the set of observations $\mathcal{O}$, and rewards from $\mathcal{R} \subseteq \mathbb{Q}$, and sends them to all the agents. Then, each agent performs actions from a limited set of actions $\mathcal{A}$ in response. An iteration or step i stands for one sequence of observation-action-reward. An observation at iteration i is denoted by o_i, while the corresponding action and reward for the same iteration are denoted by a_i and r_i respectively. The string $o_1a_1r_1o_2a_2r_2$ is an example sequence of interactions over two iterations between one agent and its environment. An illustration of the agent-environment framework is given in Figure 1. We define the multi-agent-environment framework as an extension of the above, such that $o_{i,j}, a_{i,j}$ and $r_{i,j}$ are respectively the observation, action and reward for agent π_j at iteration i. The order of interactions starts by the environment sending observations to all the agents at the same time. Then, the agents interact and perform corresponding actions, and finally the environment provides them back with rewards. For instance, the first interaction of agents

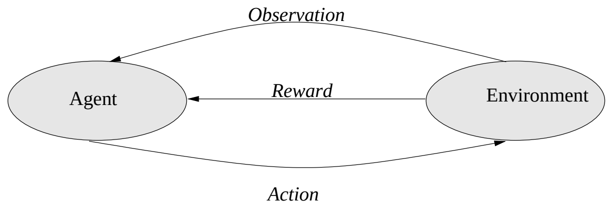


Figure 1: Agent-environment framework [30]

π_1, π_2 in the multi-agent-environment setting, denoted by $o_1a_1r_1$, is equivalent to $o_{1,1}o_{1,2}a_{1,1}a_{1,2}r_{1,1}r_{1,2}$.

4 EVALUATING INTELLIGENCE

In order to assess the performances of AI agents, whether in isolation or collectively, we needed an environment over which we can run formal intelligence tests (of measurable complexities) on artificial agents using the recently described framework. Hence, we developed in our recent work [9] an extension of the Λ environment [28, Sec. 6][22] - one of the environment classes implementing the theory behind the "Anytime Universal Intelligence Test" [22].

One of the reasons for selecting the Anytime Universal Intelligence Test and the Λ environment was because they are derived from formal and mathematical backgrounds that have been practically used to evaluate diverse kinds (including machines) of entities [26, 7, 8, 27]. More importantly, our selection embraces all of the concerns we raised in the introduction regarding the measurement of intelligence, thus providing us with a formal, anytime, dynamic and unbiased setting [22] to quantitatively evaluate the effectiveness of artificial agents.

4.1 The Λ^* (Lambda Star) Environment

We re-introduce the Λ^* (Lambda Star) environment class used in [9] which is an extension of the Λ environment [22, Sec. 6.3][28] that focuses on a restricted - but important - set of tasks in AI.

The general idea is to evaluate an agent that can perform a set of actions, by placing it in a grid of cells with two special objects, *Good* ($\oplus$) and *Evil* ($\ominus$), travelling in the space using movement patterns of measurable complexities. The rewards are defined as a function of the position of the evaluated agent with respect to the positions of $\oplus$ and $\ominus$.

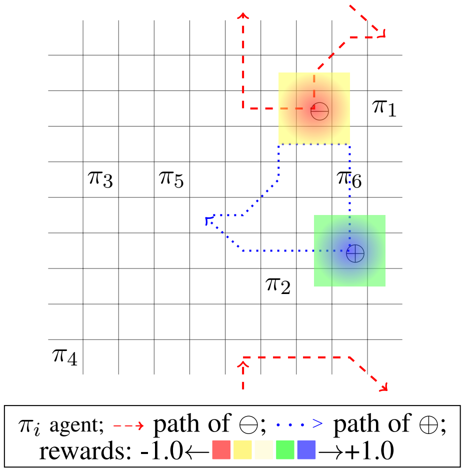
4.1.1 Structure of the test

An environment space is an m-by-n grid-world populated with objects from $\Omega = \{\pi_1, \pi_2, \ldots, \pi_x, \oplus, \ominus\}$, the finite set of objects. The set of evaluated agents $\Pi \subseteq \Omega$ is $\{\pi_1, \pi_2, \ldots, \pi_x\}$. Each element in Ω can have actions from a finite set of actions $\mathcal{A} = \{$*up-left, up, up-right, left, stay, right, down-left, down, down-right*$\}$. All objects can share the same cell at the same time except for $\oplus$ and $\ominus$ where in this case, one of them is randomly chosen to move to the intended cell while the other one keeps its old position. In the context of the agent-environment framework [30], a test episode consisting of a series of ϑ interactions $o_ia_ir_i$ such that $1 \leq i \leq \vartheta$, is modelled as follows:

1. the environment space is first initialised to an m-by-n toroidal grid-world, and populated with a subset of evaluated agents from $\Pi \subseteq \Omega$, and the two special objects $\oplus$ and $\ominus$,

2. the environment sends to each agent a description of its range of 1 *Moore neighbour* cells [17, 48] and their contents, the rewards in these cells, as an observation,
3. the agents (communicate/interact and) respond to the observations by performing an action in $\mathcal{A}$, and the special objects perform the next action in their movement pattern,
4. the environment then returns a reward to each evaluated agent based on its position (distance) with respect to the locations of the special objects,
5. this process is repeated again from point #2 until a test episode is completed, that is when $i = \vartheta$.

The Λ^* environment consists of a toroidal grid space in the sense that moving off one border makes an agent appear on the opposite one. Consequently, the distance between two agents is calculated using the surpassing rule (toroidal distance) such that, in a 5-by-5 grid space for example, the distance between cell $(1, 3)$ and $(5, 3)$ is equal to 1 cell. An illustration of the Λ^* environment is given in Figure 2.



π_1
π_3
π_5
π_6
π_2
π_4
π_i agent; $\dashrightarrow$ path of $\ominus$; $\cdots>$ path of $\oplus$;
rewards: -1.0← 🟥 🟨 ⬜ 🟩 🟦 →+1.0

Figure 2: A diagrammatic representation of a sample 10-by-10 Λ^* environment space used in [9] to evaluate the performance of (groups of interactive) artificial agents. The figure shows the objects in Ω, the paths of the two special objects and an illustration of the (positive and negative) rewards in the environment.

4.1.2 *Rewarding function*

The environment sends a reward to each evaluated agent from the set of rewards $\mathcal{R} \subseteq \mathbb{Q}$ where $-1.0 \leq \mathcal{R} \leq 1.0$. Given an agent π_j, its reward $r^i_j \in \mathcal{R}$ at some test iteration i can be calculated as:

$$r^i_j \leftarrow \frac{1}{d(\pi_j, \oplus) + 1} - \frac{1}{d(\pi_j, \ominus) + 1}$$

where $d(a, b)$ denotes the (toroidal) distance between two objects a and b. Recall that an agent does not have a full representation of the space and only receives observations of its (range of 1 Moore) neighbourhood [17, 48]. Therefore, we constrain the (positive and negative) rewards an agent receives from the environment (as a function of its position with respect to $\oplus$ and $\ominus$ respectively) as follows: the positive reward π_j receives at each iteration is calculated as $1/(d(\pi_j, \oplus) + 1)$ if $d(\pi_j, \oplus) < 2$, or 0 otherwise. Likewise its negative reward at that iterations is $-1/(d(\pi_j, \ominus) + 1)$ if $d(\pi_j, \ominus) < 2$, or 0 otherwise. Its total reward, r^i_j at iteration i, is the sum of its positive and negative rewards received at that iteration.

4.2 Algorithmic Complexity

We regard the Kolmogorov complexity [32][2] of the movement patterns of the special objects as a measure of the algorithmic complexity $K(\mu)$ of the environment μ in which they operate. For instance, a Λ^* environment of high Kolmogorov complexity is sufficiently rich and structured to generate complicated (special object) patterns/sequences of seeming randomness.

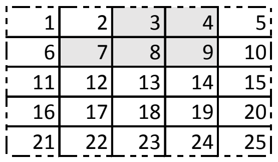
The Kolmogorov complexity [32] (Definition 1) of a string x is the length of the shortest program p that outputs x over a reference (Turing) machine U.

Definition 1 Kolmogorov Complexity

$$K_U(x) := \min_{p:\ U(p)=x} l(p)$$

where $l(p)$ denotes the length of p in bits, and $U(p)$ denotes the result of executing p on a Universal Turing machine U.

Assume, for example, that the special object $\oplus$ moves in a 5-by-5 grid space. It has an ordered (and repeating) movement pattern travelling between cells with indices: $7, 3, 4, 9$ and 8 (grayed cells appearing in Figure 3) such that, in a 25-cell grid, indices $1, 2$ and 6 correspond respectively to cells with coordinates $(1, 1)$, $(1, 2)$ and $(2, 1)$ and so on (Figure 3). Also assume that the number of time steps in one test episode $\vartheta = 20$. Following algorithmic information theory, namely Kolmogorov complexity, we consider the algorithmic complexity of the environment $K(\mu)$ in which $\oplus$ operates as the length of the shortest program that outputs the sequence 73498734987349873498 (of length ϑ). We measure the Lempel-Ziv complexity [31] of the movement patterns as an approximation to $K(\mu)$ as suggested in [31, 15].



1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure 3: A conceptual representation of a 5-by-5 grid space and its cell indices ranging from 1 to 25.

Note that, at one test episode, the movement patterns of the special objects $\oplus$ and $\ominus$ are different but (algorithmically) equally complex making sure the rewards are balanced [20]. The recurrent segment of the movement pattern is at least of length one and at most $\lfloor \vartheta/2 \rfloor$, cyclically repeated until the final iteration (ϑ) of the test.

4.3 Search Space Complexity

We measure the search space complexity $\mathcal{H}(\mu)$ as the amount of *uncertainty* in μ, expressed by Shannon's entropy [38]. Let N be

[2] The concept of Kolmogorov complexity or algorithmic information theory (AIT) is based on independent work of R. J. Solomonoff [39, 40, 41] and A. N. Kolmogorov [29] in the first half of the 1960s, shortly followed by related work by G. J. Chaitin [3, 4]. The relationship between this work and the Minimum Message Length (MML) principle (also from the 1960s) [45] is discussed in [46][44, Chapter 2 and Section 10.1][10, Sections 2 and 6].

the set of all possible states of an environment μ such that a state s_μ, is the set holding the current positions of the special objects $\{\oplus, \ominus\}$ in the m-by-n space. Thus the number of states $|N|$ increases with the increase in the space dimensions m and n, and it is equal to the number of permutation ${}^{\mathrm{m}\times\mathrm{n}}P_2 = \frac{(\mathrm{m}\times\mathrm{n})!}{(\mathrm{m}\times\mathrm{n}-2)!}$. The entropy is maximal at the beginning of the test as, from an agent's perspective, there is complete uncertainty about the current state of μ. Therefore $p(s_\mu)$ follows a uniform distribution and is equal to $1/|N|$. Using $\log_2$ as a base for our calculations, we end up with: $\mathcal{H}(\mu) = -\sum_{s_\mu \in N} p(s_\mu) \log_2 p(s_\mu) = \log_2 |N|$ bits.

Algorithmic and search space complexities could be combined into a higher level complexity measure of the whole environment. This new measure can be very useful to weight test environments used for the measurement of universal intelligence. Nonetheless, having two separate measures of complexity also means that we can quantify the individual influence of each class (or type) of complexity on the performance of agents. This approach appears to be particularly useful for evaluating the factors influencing the performance of agent collectives as these collectives can exhibit different behaviors in response to changes in the measures of each class of environment complexity [9, Section 7].

Overall, we appraise the Λ^* environment, at a minimum, as an accurate measure of the subject's ability of performing over a class of: inductive-inference, compression, and search problems, all of which are particularly related to intelligence [25, 14]. Note, however, that we will use the term *intelligence* to describe the effectiveness or accuracy of an evaluated agent over this test.

4.4 Intelligence Score

The metric of (individual agent) universal intelligence defined in [22, Definition 10] was extended into a collective intelligence metric (Definition 3) returning an average reward accumulation per-agent measure of success (Definition 2) for a group of agents Π, over a selection of Λ^* environments (Section 4.1).

Definition 2 *Given a Λ^* environment μ and a set of (isolated or interactive) agents $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ to be evaluated, the (average per-agent per-iteration) reward $\tilde{R}_{\Pi,\mu,\vartheta}$ of Π over one test episode of ϑ-iterations is calculated as: $\tilde{R}_{\Pi,\mu,\vartheta} = \frac{\sum_{j=1}^{n}\sum_{i=1}^{\vartheta} r_j^i}{n\times\vartheta}$.*

Definition 3 *The (collective) intelligence of a set of agents Π is defined as: $\Upsilon(\Pi) = \frac{1}{\omega}\sum_{\mu\in L}\tilde{R}_{\Pi,\mu,\vartheta}$, where L is a set of ω environments $\{\mu_1, \mu_2, \ldots, \mu_\omega\}$ such that $\forall \mu_t, \mu_q \in L$: $\mathcal{H}(\mu_t) = \mathcal{H}(\mu_q)$, and $\forall \mu_i \in L, K(\mu_i)$ is extracted from a range of (pattern) algorithmic complexities in $]1, K_{max}]$.*

Note the use of the same search space complexity, but different algorithmic complexities, in the intelligence measure defined in Definition 3. The reason behind this is to allow for running controlled experiments to test against the influence each class of complexity has on intelligence separately in a similar manner to [9, Sections 7.3 and 7.6].

5 IMPLEMENTATION DETAILS AND EXPERIMENTAL PROTOCOL

In this section we discuss some important test functionalities and experimental parameters, and give technical description of example agent behaviours showing how they can be practically evaluated over the Λ^* environment.

5.1 Setup and Test Parameters

The intelligence test was implemented in C++, and the source code and scripts to run experiments have been released as open-source [6], with good efforts made to facilitate their re-usability.

Once the test is compiled and run, a new experiment is initiated. The number of test episodes ω, as well as the number of iterations in each episode, for that experiment can be entered into the commandline. Setting ω to 1000 episodes (runs) usually records a very small standard deviation between the test scores[3]. The size of the environment (and thus the search space uncertainty $\mathcal{H}(\mu)$) as well as the number of agents to be evaluated can also be selected prior to each experiment. The robustness of the test scores depends on the size of the environment so it might be desirable to select a larger value of ω for larger environment spaces.

In each episode, agents are administered over different pattern complexities $K(\mu)$ automatically generated by the test, such that $K(\mu) \in [2, 23]$, where a $K(\mu)$ of 23 corresponds to, more or less, complex pattern prediction or recognition problems. Moreover, in each episode, the evaluated agents are automatically re-initialised to different spatial positions in the environment. At the end of each experiment the (intelligence) scores (in the range $[-1.0, 1.0]$) of the evaluated agents and collectives, averaged over all test episodes, are displayed on the screen and also saved to file.

Agents can be evaluated in isolation as well as collectively following the agent environment framework described in Section 3. For instance, the test provides us with three key methods implementing the (multi) agent-environment framework . Let $\tilde{\mu}$ be an instance of the test environment Λ^* and Π a set of agents to be evaluated. The methods $sendObservations(\Pi, k)$ and $sendReward(\Pi, i)$ could be invoked on $\tilde{\mu}$ at each iteration i of the test in order to send observations and rewards respectively to all agents in Π, where $k \in \mathbb{N}^+$ refers to the k^{th}-Moore neighbourhood selected as the evaluated agents' observation range. At each iteration of the test, after receiving an observation, each agent in Π invokes its own method $performAction()$ which returns a discrete action in the range $[1, 9]$, such that an action in $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ maps position-wise to $\{$*up-left, up, up-right, left, stay, right, down-left, down, down-right*$\}$. The selected action is subsequently used to update the agent's position in the environment.

5.2 Defining Agent Behaviours

We have defined an abstract class *Agent* with many declared functionalities that will come out to be essential for implementing and evaluating new agent behaviours over the Λ^* environment.

New isolated (non-interactive) agent behaviours can be introduced as (one of the) subclasses of *Agent*, providing implementations for its abstract methods as necessary. Interactive agent behaviours, on the other hand, are polymorphic classes redefining and extending the behaviour of their isolated agent's superclass.

Homogeneous collectives of interactive agents are aggregations of two or more interactive agents of the same behaviour (class). A simplified UML class diagram illustrating the relationships between isolated and collective agent behaviours is illustrated in Figure 4. Likewise, heterogeneous collectives of agents can be defined as aggregations of two or more interactive agents of different behaviours (classes). Examples of (isolated and collective) agent behaviours are described in the next two subsections.

[3] Usually a standard deviation of less than 0.001 is recorded between identical experiments.

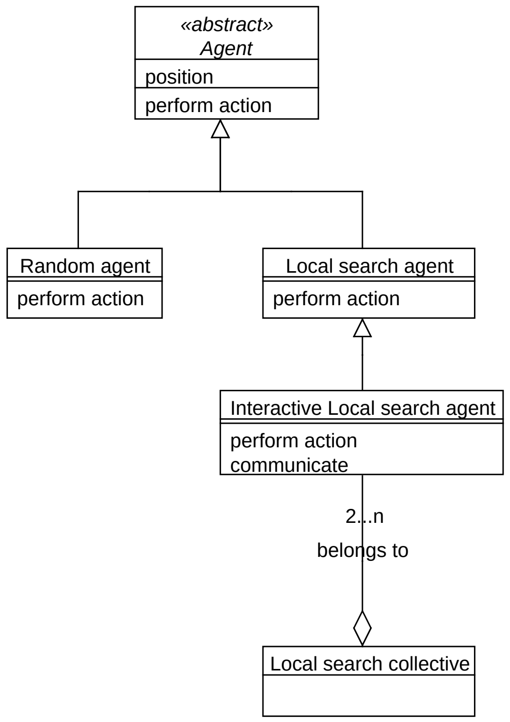


Figure 4: A simplified UML class diagram illustrating the relationships between some isolated and collective agent behaviours.

5.2.1 *Isolated agent behaviours*

In this subsection, we give a description of some agent behaviours[4] which could be evaluated over the Λ^* environment.

Local search agent: given an agent π_j, we denote by c_j^i and $r(c_j^i)$ the cell where π_j is located at iteration i, and the reward in this cell respectively. Let N_j^i and $R(N_j^i)$ denote respectively the set of neighbour cells of agent π_j (including c_j^i) at iteration i, and the reward values in these cells. $\mathtt{R}(c_j^i, a)$ is a function that returns the reward agent π_j gets after performing action $a \in \mathcal{A}$ when it is in cell c_j^i. The behaviour of a local search agent π_j at iteration i is defined as follows:

$$a_j^i \leftarrow \arg\max_{a \in \mathcal{A}} \mathtt{R}(c_j^i, a).$$

If all actions return an equivalent reward, then a random action in $\mathcal{A}$ is selected.

Q-learning agent: in this reinforcement learning behaviour, the evaluated Q-learning [47] agent learns using a state-action pair quality function, $Q : S \times \mathcal{A} \rightarrow \mathbb{R}$, in order to find the action-selection policy that maximises its rewards. Each test episode of ϑ iterations is equivalent to one training session. Because the testing environment is dynamic, we define a Q-learning state $s_i \in S$ that an agent π_j occupies at iteration i as the pair $\{c_j^i, i\}$ consisting of π_j's current cell position c_j^i and the current iteration i, thus leading to a total number of states $|S| = \mathrm{m} \times \mathrm{n} \times \vartheta$, in a m-by-n environment space, over one test episode. The Q-Learning behavior over one training session is illustrated in Algorithm 1. We use the notations from the previous (Local search agent) paragraph. After training is complete, the evaluated agent simply travels between states by performing the actions with the highest reward values recorded in its Q-table.

Algorithm 1 Q-Learning agent behavior over one training session.

```
1: Initialize: learning rate $\alpha$ and discount factor $\gamma$.
2: Begin
3:     for iteration $i \leftarrow 0$ to $\vartheta - 1$ do                    ▷ loop over iterations
4:         $s_i \leftarrow \{c_j^i, i\}$                                  ▷ set current state
5:         execute $a_j^i \leftarrow \arg\max_{a \in \mathcal{A}} Q(s_i, a)$          ▷ perform action
6:         $s_{i+1} \leftarrow \{c_j^{i+1}, i+1\}$                      ▷ set new (post-action) state
7:         $Q(s_i, a_j^i) = Q(s_i, a_j^i) + \alpha \left[\mathtt{R}(c_j^i, a_j^i) + \gamma \max_{a \in \mathcal{A}} Q(s_{i+1}, a) - Q(s_i, a_j^i)\right]$   ▷ update Q-table
8:     end for
9: End
```

Expert agent: an expert or oracle agent knows the future movements of the special object $\oplus$. At each step i of an episode this agent approaches the subsequent $i+1$ cell destination of $\oplus$ seeking maximum payoff. However, if $\oplus$ has a constant movement pattern (e.g., moves constantly to the right) pushing it away from the oracle, then the oracle will move in the opposite direction in order to intercept $\oplus$ in the upcoming test steps. Once it intercepts $\oplus$, it then continues operating using its normal behaviour.

Random agent: a random agent randomly choses an action from the finite set of actions $\mathcal{A}$ at each iteration until the end of an episode.

The scores of the random and oracle agents could be used as a baseline for the intelligence test scores of artificial agents, where a random agent is used as a lower bound on performance while the expert agent is used as an upper bound.

5.2.2 *Agent collectives*

The isolated agents could also be evaluated collectively (in groups) using a communication protocol to interact between one another. We propose a simple algorithm for enabling communication between local search agents using stigmergy [16] which is a form of indirect communication. For instance, we let local search agents induce fake rewards in the environment, thus indirectly informing neighbour agents about the proximity of the special objects. Note that fake rewards will not affect the score (real reward payoff) of the agents.

Let $\hat{R}(N_j^i)$ denote the set of fake rewards in the neighbour cells of agent π_j (including c_j^i) at iteration i, and $\hat{\mathtt{R}}(c_j^i, a)$ is a function returning the fake reward agent π_j receives after performing action $a \in \mathcal{A}$ when it is in cell c_j^i at iteration i. Fake rewards are induced in the environment according to Algorithm 2. Each agent proceeds by selecting an action by relying on fake rewards this time instead of the real rewards, as follows: $a_j^i \leftarrow \arg\max_{a \in \mathcal{A}} \hat{\mathtt{R}}(c_j^i, a)$. If all actions

Algorithm 2 Stigmergic or indirect communication: fake reward generation over one iteration i of the test.

```
1: Input: $\Pi$ (set of evaluated agents), $0 < \gamma < 1$ (fake reward discounting
   factor), a test iteration $i$.
2: Initialize: $\forall \pi_j \in \Pi$: $\hat{R}(N_j^i) \leftarrow 0.0$.
3: Begin
4:     for j $\leftarrow 1$ to $|\Pi|$ do                              ▷ loop over agents
5:         $r^{max} \leftarrow \max R(N_j^i)$
6:         $r^{min} \leftarrow \min R(N_j^i)$
7:         $\hat{r} \leftarrow \gamma \cdot r^{max} + (1 - \gamma) \cdot r^{min}$
8:         $\hat{R}(N_j^i) \leftarrow R(N_j^i) + \hat{r}$
9:     end for
10: End
```

[4] Several agent behaviours (isolated and collectives) other than those discussed in this paper have also been implemented and are made available in [6] for both testing and modification.

are equally rewarding, then a random action is selected. Thereupon, we expect local search agents using stigmergy to form non-strategic coalitions after a few iterations of the test as a result of tracing the most elevated fake rewards in the environment.

In the case of Q-learning collectives, the agents in the collective could share and update a common Q-table, and thus all learn and coordinate simultaneously.

5.3 Modularity and Code Re-use

We have provided a large set of functionalities which might come in handy when amending and extending the current scope of the test, and for defining new agent behaviours to be evaluated. These functionalities can be found in the utility class *General* in [6] under the directory `/src/General.cpp`. Moreover, we have used UnitTest++ [33], a lightweight unit testing framework for C++ over Windows, in order to allow for easy defect isolation, assist in validating existing and newly implemented functionality, and encourage code review.

5.4 Experimental Demonstration

An example of an executable test experiment can be found in the *main* method in [6]. Similar types of experiment were conducted in our previous work [9] in order to identify and analyse factors influencing the intelligence of agent collectives. For instance, using the Λ^* environment, one could evaluate several types of multi-agent systems and quantitatively measure how:

- the complexity of the environment (its uncertainty and algorithmic complexity),
- the communication protocol used between the agents,
- the interaction time,
- and the agents' individual intelligence

all reflect (individually but also jointly) on the collective performance of the system. This can be easily achieved by running a series of controlled experiments in which the values (of one or more) of the above factors are altered.

6 ALTERNATIVE ENVIRONMENTS AND FURTHER CONSIDERATIONS

As mentioned in Section 4.1, the Λ^* (Lambda Star) environment focuses on a restricted set of canonical tasks particularly relevant to the intelligence of AI agents. Nonetheless, the generalisation of these canonical tasks does not account for a range of multiagent problems. In particular, the tasks to perform in the Λ^* environment are a nice abstraction of two problems in the literature (among others): searching for a moving target while avoiding injury, and nest selection when there is one and only one best nest. But these tasks do not cover other important multi-agent problems like those that require coordination (e.g., lifting and moving a table).

6.1 Measuring Multi-agent Coordination

Coordination is an important feature in multi-agent systems which has a high influence on their performance. Measuring coordination between interactive agents can be a difficult task. The scope of the Λ^* (Lambda Star) environment does not currently account for the measurement of coordination between agents, but we are considering extensions to assess this. For instance, problems that require coordination could have been evaluated if the payoff received from the *Good* object $\oplus$ (Section 4.1) had only occurred if two or more agents were in its neighborhood.

Another interesting extension to the test setting is to enable the environment to respond to the agent's behaviour and actions. Testing can be performed in an even more heterogeneous setting where the agents don't have the same reward function and/or actions and observations, and to give more attention or weight to the agent's learning (ability), which is an important aspect of intelligence.

Moreover, other properties like (environment) coverage could be evaluated by dispersing agents in the space to monitor what is happening in the environment (e.g., monitoring which neighborhoods are/are not explored by the agents).

6.2 Fitness Landscapes

Lambda Star (Λ^*) is one of many environments which can be used to evaluate artificial agents. A famous problem in AI is to evaluate the performance of artificial agents over fitness landscapes consisting of many local optima but only one global optimum. The landscapes reflect evaluations of some fitness or utilisation function over a set of candidate solutions. Adaptive landscapes can be considered where the underlying fitness evolves or changes over time. We have implemented a performance test based on the abovementioned problem description (by extending the Λ^* environment) and further designed a simulation depicting the behavior of (co-operative) artificial agents exploring a landscape over a period of time. The motivation is to assess the trade-off between exploration and exploitation in a reinforcement learning setting, and investigate the influences of this trade-off on the agents' payoffs in a multiple candidate solution space or environment. A screen-shot from the early stages of that simulation is given in Figure 5.

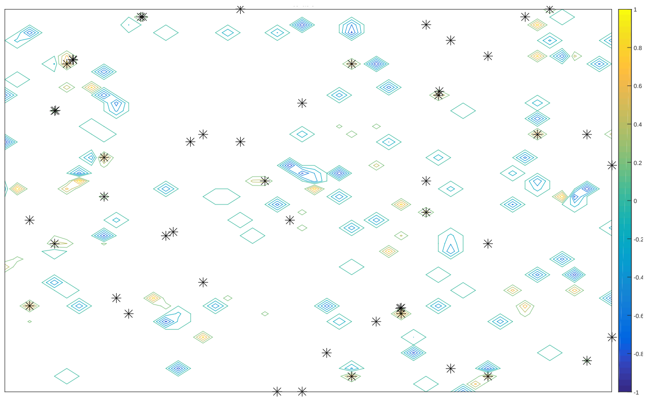


Figure 5: A screen-shot from the early stages of a simulation of a fitness landscape with many local optima but only one global optimum. Colors (and their different intensities showing in the right-hand side color-bar) represent fitness (ranging between $[-1.0, 1.0]$), or the quality of the landscape, at different spatial positions. The black stars represent agents navigating or searching the landscape.

6.3 Further Thoughts on Robust Intelligence Tests

The same way collective intelligence can emerge between artificial agents (due, for example, to the wisdom of the crowd [42], information sharing, reduction in entropy, etc.), pluralistic ignorance [37] is also a common phenomenon observed in many social settings which can occur between rational agents. Robust intelligence tests should be able to detect such a phenomenon. For instance, the field of game theory has highlighted several scenarios where cooperation between agents does not leads to an optimal payoff (e.g., the famous prisoner's dilemma [35]). A robust intelligence test should be general enough to reflect and evaluate such scenarios.

Other multi-agent phenomena witnessed in various social settings reflect how agents acting individually might perform adversely to the common good, and thus deplete their available resources as a consequence of their collective behavior. A robust intelligence test should allow for a quantitative assessment of *the tragedy of the commons* [18] phenomena occurring in multi-agent scenarios.

7 CONCLUSIONS

This paper provides a technical description of the design and implementation of the Λ^* environment which can be used to evaluate general purpose AI agents both in isolation and collectively. The high-level evaluation architecture, based on an agent-environment framework, is discussed. The source code and scripts to run experiments have been released as open-source, and instructions on how to administer the test to existing and new artificial agents have been outlined and supported by examples.

We have also proposed and discussed some alternative testing environments that might be useful to quantify the performance of artificial agents. We further raised some arguments and considerations (in connection with pluralistic ignorance and the tragedy of the commons phenomena) that are relevant to the robustness of multi-agent performance tests.

Having presented the above, we encourage people in the AI community to evaluate new, more advanced, types of heuristics and algorithms over the Λ^* environment, and also extend its scope to include new functionality. Multi-agent systems can now be assessed using various interaction and communication protocols. This indicates that the performance of agent collectives can be quantitatively evaluated and compared to that of individual agents. This might help answer many open questions in AI regarding the emergence of collective intelligence in artificial systems.

REFERENCES

[1] Zeungnam Bien, Won-Chul Bang, Do-Yoon Kim, and Jeong-Su Han, 'Machine intelligence quotient: its measurements and applications', *Fuzzy Sets and Systems*, **127**(1), 3 – 16, (2002).

[2] Rodney A. Brooks, 'Intelligence without reason', in *Proc. of the 1991 International Joint Conference on Artificial Intelligence*, pp. 569–595, (1991).

[3] Gregory J. Chaitin, 'On the length of programs for computing finite binary sequences', *Journal of the ACM (JACM)*, **13**(4), 547–569, (1966).

[4] Gregory J. Chaitin, 'On the length of programs for computing finite binary sequences: statistical considerations', *Journal of the ACM (JACM)*, **16**(1), 145–159, (1969).

[5] Gregory J. Chaitin, 'Godel's theorem and information', *International Journal of Theoretical Physics*, **21**(12), 941–954, (1982).

[6] Nader Chmait. The Lambda Star intelligence test code-base. `https://github.com/nader-chmait/LambdaStar.git`, 2016.

[7] Nader Chmait, David L. Dowe, David G. Green, and Yuan-Fang Li, 'Observation, communication and intelligence in agent-based systems', in *Proc. 8th Int. Conf. Artificial General Intelligence, Berlin, Germany*, volume 9205 of *Lecture Notes in Artificial Intelligence (LNAI)*, pp. 50–59. Springer, (Jul 2015).

[8] Nader Chmait, David L. Dowe, David G. Green, Yuan-Fang Li, and Javier Insa-Cabrera, 'Measuring universal intelligence in agent-based systems using the anytime intelligence test', Technical Report 2015/279, FIT, Clayton, Monash University, (2015).

[9] Nader Chmait, David L. Dowe, Yuan-Fang Li, David G. Green, and Javier Insa-Cabrera, 'Factors of collective intelligence: How smart are agent collectives?', in *Proc. of 22nd European Conference on Artificial Intelligence (ECAI)*, (2016).

[10] David L. Dowe, 'MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness', in *Handbook of the Philosophy of Science - Volume 7: Philosophy of Statistics*, ed., P. S. Bandyopadhyay and M. R. Forster, pp. 901–982. Elsevier, (2011).

[11] David L. Dowe and Alan R. Hajek, 'A computational extension to the Turing Test', *Proc. 4th Conf. of the Australasian Cognitive Science Society, University of Newcastle, NSW, Australia*, (1997).

[12] David L. Dowe and Alan R. Hajek, 'A computational extension to the Turing Test', *Technical Report #97/322, Dept Computer Science, Monash University, Melbourne, Australia*, (1997).

[13] David L. Dowe and Alan R. Hajek, 'A non-behavioural, computational extension to the Turing Test', in *International conference on computational intelligence & multimedia applications (ICCIMA'98), Gippsland, Australia*, pp. 101–106, (1998).

[14] David L. Dowe, José Hernández-Orallo, and Paramjit K. Das, 'Compression and intelligence: Social environments and communication', in *Proc. 4th Int. Conf. on AGI*, pp. 204–211, Berlin, (2011). Springer.

[15] Scott C. Evans, John E. Hershey, and Gary Saulnier, 'Kolmogorov complexity estimation and analysis', in *6th World Conf. on Systemics, Cybernetics and Informatics*, (2002).

[16] Plerre-P. Grassé, 'La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs', *Insectes sociaux*, **6**(1), 41–80, (1959).

[17] Lawrence Gray, 'A mathematician looks at Wolfram's new kind of science', *Notices of the American Mathematical Society*, **50**(2), 200–211, (2003).

[18] Garrett Hardin, 'The tragedy of the commons', *Science*, **162**(3859), pp. 1243–1248, (1968).

[19] José Hernández-Orallo, 'Beyond the Turing Test', *J. of Logic, Lang. and Inf.*, **9**(4), 447–466, (October 2000).

[20] José Hernández-Orallo. A (hopefully) unbiased universal environment class for measuring intelligence of biological and artificial systems, 2010.

[21] José Hernández-Orallo, *The Measure of All Minds: Evaluating Natural and Artificial Intelligence*, Cambridge University Press, 2016. to appear.

[22] José Hernández-Orallo and David L. Dowe, 'Measuring universal intelligence: Towards an anytime intelligence test', *Artificial Intelligence*, **174**(18), 1508–1539, (December 2010).

[23] José Hernández-Orallo and David L. Dowe, 'On potential cognitive abilities in the machine kingdom.', *Minds and Machines*, **23**(2), 179–210, (2013).

[24] José Hernández-Orallo, Fernando Martinez-Plumed, Ute Schmid, Michael Siebers, and David L. Dowe, 'Computer models solving intelligence test problems: Progress and implications', *Artificial Intelligence*, **230**, 74 – 107, (2016).

[25] José Hernández-Orallo and Neus Minaya-Collado, 'A formal definition of intelligence based on an intensional variant of Kolmogorov complexity', in *Proc. of the Int. Symposium of EIS*, pp. 146–163. ICSC Press, (1998).

[26] Javier Insa-Cabrera, José-Luis Benacloch-Ayuso, and José Hernández-Orallo, 'On measuring social intelligence: Experiments on competition and cooperation', in *Proc. 5th Conf. on AGI*, eds., Joscha Bach, Ben Goertzel, and Matthew Iklé, volume 7716 of *LNCS*, pp. 126–135. Springer, (2012).

[27] Javier Insa-Cabrera, David L. Dowe, Sergio España-Cubillo, M. Victoria Hernandez-Lloreda, and Jose Hernandez-Orallo, 'Comparing humans and AI agents.', in *AGI*, volume 6830 of *LNCS*, pp. 122–132. Springer, (2011).

[28] Javier Insa-Cabrera, José Hernández-Orallo, David L. Dowe, Sergio España, and M Hernández-Lloreda, 'The ANYNT project intelligence test: Lambda-one', in *AISB/IACAP 2012 Symposium "Revisiting Turing and his Test"*, pp. 20–27, (2012).

[29] Andrei N. Kolmogorov, 'Three approaches to the quantitative definition of information', *Problems of information transmission*, **1**(1), 1–7,

(1965).

[30] Shane Legg and Marcus Hutter, 'Universal intelligence: A definition of machine intelligence', *Minds and Machines*, **17**(4), 391–444, (2007).

[31] Abraham Lempel and Jacob Ziv, 'On the Complexity of Finite Sequences', *Information Theory, IEEE Transactions on*, **22**(1), 75–81, (January 1976).

[32] Ming Li and Paul Vitányi, *An introduction to Kolmogorov complexity and its applications (3rd ed.)*, Springer-Verlag New York, Inc., 2008.

[33] Noel Llopis and Charles Nicholson. UnitTest++ testing framework. `https://github.com/unittest-cpp/unittest-cpp`, last accessed, April 2016.

[34] Graham Oppy and David L. Dowe, 'The Turing Test', in *Stanford Encyclopedia of Philosophy*, ed., Edward N. Zalta. Stanford University, (2011).

[35] William Poundstone, *Prisoner's dilemma*, Anchor, 2011.

[36] Pritika Sanghi and David L. Dowe, 'A computer program capable of passing I.Q. tests', in *Proc. of the Joint International Conference on Cognitive Science, 4th ICCS International Conference on Cognitive Science & 7th ASCS Australasian Society for Cognitive Science (ICCS/ASCS-2003)*, ed., P. P. Slezak, pp. 570–575, Sydney, Australia, (13-17 July 2003).

[37] Fatima B. Seeme and David G. Green, 'Pluralistic ignorance: Emergence and hypotheses testing in a multi-agent system', in *Proc. of IEEE International Joint Conference on Neural Networks (IJCNN)*, (2016).

[38] Claude E. Shannon, 'A mathematical theory of communication', *Bell System Technical Journal, The*, **27**(3), 379–423, (July 1948).

[39] Ray J. Solomonoff, 'A preliminary report on a general theory of inductive inference (report ztb-138)', *Cambridge, MA: Zator Co*, **131**, (1960).

[40] Ray J. Solomonoff, 'A formal theory of inductive inference. part I', *Information and control*, **7**(1), 1–22, (1964).

[41] Ray J. Solomonoff, 'A formal theory of inductive inference. part II', *Information and control*, **7**(2), 224–254, (1964).

[42] James Surowiecki, *The Wisdom of Crowds*, Anchor, 2005.

[43] Alan M. Turing, 'Computing machinery and intelligence', *Mind*, **59**, 433–460, (1950).

[44] Christopher S. Wallace, *Statistical and inductive inference by minimum message length*, Springer Science & Business Media, 2005.

[45] Christopher S. Wallace and David M. Boulton, 'An information measure for classification', *The Computer Journal*, **11**(2), 185–194, (1968).

[46] Christopher S. Wallace and David L. Dowe, 'Minimum message length and Kolmogorov complexity', *The Computer Journal*, **42**(4), 270–283, (1999). Special issue on Kolmogorov complexity.

[47] Christopher J. C. H. Watkins and Peter Dayan, 'Technical note: Q-learning', *Mach. Learn.*, **8**(3-4), 279–292, (May 1992).

[48] Eric W. Weisstein. Moore neighborhood, from mathworld - a Wolfram web resource. `http://mathworld.wolfram.com/MooreNeighborhood.html`, 2015. Last accessed: 2015-11-10.

PAGI World: A Physically Realistic, General-Purpose Simulation Environment for Developmental AI Systems

John Licato[1] **and Selmer Bringsjord**[2]

Analogical Constructivism and Reasoning Lab[1]
Indiana University/Purdue University - Fort Wayne

Rensselaer AI and Reasoning Lab[2]
Rensselaer Polytechnic Institute

Abstract. There has long been a need for a simulation environment rich enough to support the development of an AI system sufficiently knowledgeable about physical causality to pass certain tests of Psychometric Artificial Intelligence (PAI) and Psychometric Artificial General Intelligence (PAGI). In this article, we present a simulation environment, PAGI World, which is: cross-platform (as it can be run on all major operating systems); open-source (and thus completely free of charge to use); able to work with AI systems written in almost any programming language; as agnostic as possible regarding which AI approach is used; and easy to set up and get started with. It is our hope that PAGI World will give rise to AI systems that develop truly rich knowledge and representation about how to interact with the world, and will allow AI researchers to test their already-developed systems without the additional overhead of developing a simulation environment of their own. After clarifying both PAI and PAGI, we summarize arguments that there is great need for a simulation environment like PAGI World. We present multiple examples of already-available PAI and PAGI tasks in PAGI World, covering a wide range of research areas of interest to the general-purpose AI community.

1 Introduction

Toward the end of his long and extremely distinguished career, Jean Piaget began to name and concretely describe some mechanisms he believed were responsible for the emergence of many features of mature cognition: formal reasoning, an understanding of causality, and analogical ability were among these features, along with many others [26, 25, 28]. Piaget had long suspected that these features and the concepts they relied on were constructed by the child using simpler schemas acquired through interaction with the physical world, at least since (Piaget and Inhelder 1958). Thus the role that the world plays in shaping the constructs and abilities of the child, which informs the related question of how much AI can progress without having a real-world-like environment, has been a cornerstone issue in AI for some time now [10, 11, 18].

But modeling these Piagetian beliefs is, to this day, an unmet goal that has existed at least since (Drescher 1991). Such modeling is a dream of computational cognitive modelers, but, perhaps more specifically, is a goal of the field of developmental AI. This is the field which attempts to show how, using an agent endowed with minimal innate capacities embedded in a sufficiently rich environment, higher-level cognitive abilities can emerge [17]. These abilities may include logico-mathematical reasoning, an understanding of causality, robust analogical reasoning, and others. Furthermore, work in developmental AI systems strives to show that the emergence of such abilities could be reflective of the way they develop in humans, whether this is in the pattern predicted by Piaget's stage theories or not.

This paper describes a task-centered, physically realistic simulation environment that we have developed to simultaneously address a set of challenges in evaluating AGIs. We motivate PAGI World in Section 2, introduce PAGI World in Section 3, and outline examples of a wide variety of tasks in PAGI World in Section 4.

2 Motivations

Here we summarize three categories of motivations for PAGI World, particularly of interest to those wishing to evaluate artificially-general intelligence (AGI). **C1** - **C6** specify conditions for a *sufficiently rich* simulation environment. The Tailorability Concern (Section 2.2) deals with the way in which an AGI acquires and constructs its knowledge. Finally, Section 2.3 puts forth our belief that an AGI's knowledge should be *expressive*, in the sense of logical expressivity.

2.1 Guerin's Conditions

Frank Guerin [17], in his recent survey of the developmental AI field, concluded that current systems were lacking in several key areas. Guerin then suggested that a major reason (arguably the most important reason) why the field has the shortcomings he described, is the absence of a suitable simulation environment. Current simulation environments used by developmental-AI projects were missing several key features, and Guerin described some conditions that would need to be met by simulation environments in order to address this problem. We refer to the most important of these conditions as **C1**, **C2**, and **C3**. A sufficiently rich simulation environment for developmental AI should, at a minimum:

C1 be rich enough to provide knowledge that would bootstrap understanding of concepts rooted in physical relationships; e.g.: inside vs. outside, large vs. strong, etc.

C2 allow for the modeling and acquisition of spatial knowledge, which Guerin notes is widely regarded to be a foundational domain of knowledge acquisition, through interaction with the world.

C3 support the creation and maintenance of knowledge the agent can verify itself.

[21] introduced a few additional conditions:

C4 be rich enough to provide much of the sensory-level information that an agent in the real world would have access to.

C5 allow for testing of a virtually unlimited variety of tasks, whether these are tasks testing low-level implicit knowledge, high-level explicit knowledge, or any of the other areas required by Psychometric Artificial General Intelligence (PAGI). Ideally, such a system would support the easy creation of new tasks and environments without requiring a massive programming effort.

C6 provide pragmatic features enabling tasks to be attempted by researchers using different types of systems and different theoretical approaches, thus enabling these different approaches to be directly compared with each other.

These conditions were elaborated on and defended in [21], so we will not do so here. A common theme running through all six conditions is that what is lacking from current microworlds is a physically realistic environment—one in which the agent can acquire, develop, and test its concepts. But the concerns raised by Guerin are not only of interest to the field of Developmental AI; in point of fact, *all* of AI can benefit by addressing them. For example, **C1** is extremely important for cognitive models of analogy, as they struggle to overcome what has been called the *Tailorability Concern* (TC)[16, 22].

2.2 The Tailorability Concern

TC, in essence, is the concern that models of analogy (though this can be applied to all cognitive architectures in general) work almost exclusively with manually constructed knowledge representations, using toy examples often *tailor-made* to display some limited-scope ability. Licato et al. ([22]) argue that overcoming TC is necessary to advance the fields of analogy and cognitive architectures, by developing a set of conditions that must be met in order to claim victory over TC:

> **TCA$_3$** A computational system of analogy answers TC if and only if given no more than either
>
> - unstructured textual and/or visual data, or
> - a large, pre-existing database,
>
> and minimal input, it is able to consistently produce *useful* analogies and demonstrate stability through a variety of input forms and domains.

According to **TCA$_3$**, then, good performance on the part of a cognitive agent on a sufficiently large knowledge-base from which source analogs could be drawn is required to answer TC. An agent interacting in the sort of microworld called for by Guerin might ideally be able to acquire such source analogs by simply interacting with its environment.

C1 and TC together require that the microworld itself is what provides the knowledge drawn upon to construct concepts of basic physical relationships, not manually constructed source analogs or fully explicit logical theories. **C2** expands on **C1** by requiring that this knowledge of physical relationships not be static, but rather should allow for an agent in the world to learn through interaction. The idea that children learn by initiating interactions with the world based on their (often incomplete) conceptions of reality—in a manner that resembles scientific experimentation—was championed by Piaget and later, Piaget-influenced work [2, 29, 27, 39].

Following **TCA$_3$**, another formulation of the Tailorability Concern and recommendation for how to surpass it was also presented in [22]:

> **TCA$_4$** A computational system $\mathcal{A}$ for analogy generation answers TC if and only if, given as input no more than either
>
> - unstructured textual and/or visual data, or
> - a vast, pre-existing database not significantly pre-engineered ahead of time by humans for any particular tests of $\mathcal{A}$,
>
> is—in keeping with aforementioned *Psychometric AI*—able to consistently generate analogies that enable $\mathcal{A}$ to perform *provably well* on precisely defined tests of cognitive ability and skill.

TCA$_4$ ties TC to Artificial General Intelligence (AGI) by introducing the concept of *Psychometric AI* (PAI) [3, 9]. PAI sees good performance on well-established tests of intelligence as a solid indicator of progress in AI. Some may note that most intelligence tests fail to capture human-level skills such as creativity and real-time problem solving; therefore, related to PAI is Psychometric Artificial *General* Intelligence (PAGI) [6]. For example, one test of PAGI is Bringsjord and Licato's (2012) *Piaget-MacGyver Room*, in which an agent is inside a room with certain items and a task to be performed. The agent must achieve the task using some combination of the items in the room (or using none of them, if possible). Depending on the task, the solutions may require using the items in unusual ways, as viewers of the MacGyver television series may remember. We describe several example Piaget-MacGyver Rooms in PAGI World in Section 4.[1]

2.3 Expressivity

Even after satisfying TC, it would be difficult to claim an AGI is truly general-intelligent unless its knowledge satisfies a certain degree of *logical expressivity*. By this, we do not mean that an AGI must possess a Gödel-like mastery of formal logic. Rather, the term refers to the well-established hierarchy of expressivity in formal theories. For example, a formal theory equivalent in expressivity to first-order logic (FOL) can express anything that one equivalent to propositional calculus (PC) can express. The converse is not true; something like "all men are mortal" simply cannot be expressed in a quantifier-free logic like PC, since the ability to take any possible man m and apply the statement "all men are mortal" to deduce that m is mortal is only possible with machinery that treats quantified variables *as variables that can be quantified over*.

Logical expressivity, then, is a real restriction on any formal theory's ability to express properties.[2] But the use of terminology from mathematical logic should not obscure the more general fact that logical expressivity is really a restriction on *any* system whatsoever that

[1] Note that although we have adopted "PAGI World" as the name of our simulation environment in order to reflect the fact that it is designed to support many types of PAGI tests (including variants of the Piaget-MacGyver Room, as we describe below), PAI tests are just as easily implementable in PAGI World.

[2] See [33] for a good initial definition of what it means to express properties in formal theories.

can be described using rules for producing new behaviors, actions, knowledge, or structures. All AI systems in history are no exception; once we formally describe the set of rules that govern that AI system, those rules fall under some level of logical expressivity, and whatever that level is limits what that system can ultimately do.

Furthermore, FOL's expressivity is not enough for general intelligence. Humans routinely reason over sets, analogies, the beliefs, desires, and intentions of others, and so on. Such concepts require logics significantly more expressive than FOL: second-order logic, epistemic/modal logics, even third-order logic in some cases [7]. If an AGI is to truly be as general-purpose a reasoner as the typical human, a high level of expressivity is needed.

For the first time, we present here a conjecture[3] encapsulating this view:

AGI>FOL. No system can claim to be an AGI unless its knowledge is at least more logically expressive than first-order logic.

A very high-level proof sketch of the above is as follows:

1. a concept $\mathscr{C}$ is accurately captured in a system $\mathcal{S}$ only if that system can, at a minimum, produce any actions, inferences, behaviors, or knowledge structures that would be expected of a system capturing $\mathscr{C}$.
2. A system cannot thus fully capture a concept $\mathscr{C}$ if its knowledge representation is below the level of logical expressivity required for $\mathscr{C}$.
3. There are many concepts required for AGI which are at a level of logical expressivity higher than FOL.
4. Thus, no artificial system with an expressivity at the level of FOL or lower can be an AGI.

We omit many details here, but the argument presented is at the core of a more encompassing argument for expressivity in AGI systems, currently under development. For our present purposes, suffice it to say that simulation environments which restrict the expressivity level of the knowledge of the agents which can use the environments to, or below that of FOL, can not hope to see the creation of fully general intelligence. PAGI World avoids that by placing no restrictions on the form of knowledge used by its artificial agents.

3 Introducing PAGI World

Condition **C6** is the most practicality-oriented, reflecting both Guerin's (2011) and our own inclination to believe that an effective way to compare AI and AGI methodologies would be to see how they perform on the same tasks, implemented on the same systems. But few such tasks and systems exist, and therefore before we describe PAGI World, it may be helpful to take a step back and look at our project in a broader view.

3.1 Why Isn't Such a System Already Available?

Given its potential benefit to the field as a whole, why does such an environment not currently exist, and do any of the roadblocks currently in the way affect the plausibility of our current project?

[3] The authors believe strongly in the truth of **AGI>FOL**, and ultimately hope to elevate it to the status of a theorem. However, as the present paper's scope permits only a loose proof sketch, we present it here as merely a conjecture.

3.1.1 Technical Hurdles

One potential roadblock is obvious: programming a realistic physics simulation is *hard*. Some of this difficulty is reduced by working with a 2D, rather than a 3D, environment. Although some software libraries have previously been available for 2D physics simulations, they have often been very language-specific and somewhat difficult to configure.

Secondly, even if one were to stick with a 2D physics library and commit to it, substantial development resources would be needed to enable the resulting simulation to run on more than one major operating system. Furthermore, even if *that* problem is somehow addressed, there is a vast diversity of languages that AI researchers prefer to use: Python, LISP (in various dialects, each with their own passionate proponents), C++, etc. All of these technical issues tend to reduce how willing researchers are to adopt particular simulation environments.

Fortunately, all of the above problems can be solved with a single design choice. Unity, a free game-development engine, has recently released a 2D feature set,[4] which comes with a 2D physics model that is extremely easy to work with. Furthermore, Unity allows for simultaneous compilation to all major operating systems, so that developers only have to write one version of the program, and it is trivial to release versions for Mac OS, Windows, and Linux. Because Unity produces self-contained executables, very little to no setup is required by the end users.

Finally, because Unity allows scripting in C#, we were able to write an interface for AI systems that communicates with PAGI World through TCP/IP sockets. This means that AI scripts can be written in *virtually any* programming language, so long as the language supports port communication.

3.1.2 Theoretical Hurdles

Unity conveniently helps to remove many of the technical roadblocks that have previously blocked the development of simulation environments that can be widely adopted. But there are also theoretical roadblocks; these are problems pertaining to the generality vs. work-required tradeoff. For example, if a simulation environment is too specifically tailored to a certain task, then not only can systems eventually be written to achieve that particular task and nothing else, but the simulation environment quickly becomes less useful once the task is solved. On the other hand, if the system is too general (e.g. if a researcher decides to start from scratch with nothing but Unity), then the researcher must devote too much time and energy to developing a new simulation environment for each project, rather than spending time on the AI itself.

PAGI World was designed with this tradeoff in mind. A *task* in PAGI World might be thought of as a Piaget-MacGyver Room with a configuration of objects. Users can, at run-time, open an object menu (Figure 1) and select from a variety of pre-defined world objects, such as walls made of different materials (and thus different weights, temperatures, and friction coefficients), smaller objects like food or poisonous items, functional items like buttons, water dispensers, switches, and more. The list of available world objects will frequently be expanding and new world objects will be importable into tasks without having to recreate tasks with each update. Perhaps most importantly, tasks can be saved and loaded, so that as new PAI/PAGI experiments are designed, new tasks can be created by

[4] In fact, the blog post making the announcement of the 2D feature set was dated November 12, 2013.

anyone. Section 4 illustrates the wide variety of tasks that can be created with such a system.

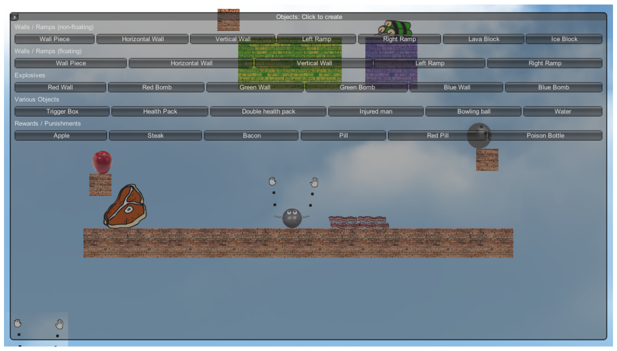


Figure 1: PAGI World With the Object Menu Visible

3.1.3 *Other Simulation Environments*

There have been some notable attempts to provide simulation environments for AI systems, particularly those inspired by the Developmental-AI approach. For example, in [12] Bruce created a Developmental-AI testbed by updating an older version created by Frank Guerin.

Although some of the present paper's authors are sympathetic to the power of Piagetian schemas and the AI systems derived from Piaget's theories, Bruce's system is tightly coupled with a particular cognitive architecture (presented in the same paper) that uses schema-based AI systems, whereas PAGI World, as we have said, is agnostic about what AI approach is used. It is unclear how easy or difficult it would be to adapt arbitrary cognitive architectures to work with their simulation environment.

They used the JBox2D library for their physics engine, which, according to [12], was poorly documented and difficult to work with (e.g., implementing a method to detect when the robot hand touched an object took markedly longer than they planned due to a lack of documentation for JBox2D). Although a newer version of JBox2D became available afterwards, implementing the new version requires the simulation programmer to manually update the relevant code, whereas updates to the Unity 2D physics engine automatically propagate to PAGI World, without any code changes on our part.

3.1.4 *Drescher's Simulation*

In [14], Drescher proposes an early microworld in which an agent, making use of a primitive form of Piagetian schemas, explores the world and learned about the objects with which it interacted. Although this was a promising start, after its initial success it was not developed further, nor was any significant effort made by other researchers to pick up on Drescher's work, as far as we are aware (only one small-scale re-implementation of Drescher's work exists, e.g. [13]).

Drescher's microworld consists of a 2D scene divided into a grid that limits the granularity of all other elements in the microworld. Inside this microworld are objects that take up discrete areas of the grid and contain visual and tactile properties, in the form of numerical vectors with arbitrarily chosen values.

Most importantly, the microworld contains a single robot-like agent with a single hand that can move in a 3-cell $\times$ 3-cell region relative to the part of the robot's body considered to be its "eye." If the hand object is adjacent to an object in the world (including the robot's own body), a four-dimensional vector containing tactile information is returned to the agent. The body has tactile sensors as well, though they do not return tactile information as detailed as that returned by the tactile sensors of the hand.

Visual information is available as well, in the form of a visual field whose position is defined relative to the robot's body. A smaller region within the visual field, called the *foveal region*, represents the area within the visual field where the robot is currently looking. The foveal region returns vectors representing visual information, and the cells in the visual field not in the foveal region also return visual information, but with lower detail.

Perhaps one of the most interesting features of Drescher's microworld is that the robot can only interact directly with the world by sending a set of predefined "built-in actions." Although the internal schema mechanism of the robot may learn to represent actions as richer and more complicated, ultimately what is sent to the simulation environment is always extremely low-level. Likewise, the information provided to the robot is always extremely low-level. The task of identifying and naming objects in the world—and even of knowing that objects in the world consistently exist!—is up to the learning mechanism the robot utilizes.

The fact that the learning and control system of the artificial agent can be developed almost completely independently of the features of the world itself, is one of the primary reasons why Drescher's microworld is appealing, and was selected as a starting point for PAGI World. PAGI World departs from, and has innovated beyond, Drescher's microworld in several key areas:

- **Agnosticism re. the AI method used**. Whereas Drescher's microworld was created for the sole purpose of testing his Piagetian schema-learning mechanism, we have designed the world, program, and interfaces so that as wide a variety as possible of AI techniques can be productively and easily used.
- **Optional mid-level input**. Related to the previous point, we realize that some researchers simply won't want to translate vector input for every piece of tactile or visual information they come across, and so we offer the option for the agent to directly receive the name of the object upon touching or viewing it.
- **Granularity**. The granularity of our world is dramatically finer; consider the increase in size of the visual field: Drescher's was an area of 7-$\times$-7 cells with one visual sensor per cell. We have improved the visual area to span a 450-$\times$-300 unit area, with each visual sensor spaced 15 units from its nearest neighbor (each unit roughly corresponds to a screen pixel).
- **Vision system**. In addition to having a wider visual field, ours has no foveal region, because the tasks we design require a visual field large enough to observe multiple objects at once. Certainly it is plausible that rapid eye movements can account for this ability in human beings, but our initial investigations found it to have too little theoretical benefit compared to how difficult it made working with the system.
- **Hands**. We have given the robot two hands instead of one, each with a similar range of motion, but with different distances (relative to the body) that each can reach. Although the simulation world is 2D, the hands exist on a separate layer that floats "above" objects in the world, analogously to a mouse cursor in any major operating system. The hands can grip and move objects they

are floating over (just as one might click and drag an object in Windows or MacOS), provided the objects are not too heavy or otherwise held down.

- **Realistic Physics**. Certainly a very important improvement we introduce is the aforementioned realistic physics provided by Unity 2D.
- **Focus on a wide breadth of tasks**. Although Drescher's microworld was a start in the right direction, we feel that it did not quite make enough of a push to be considered a simulation environment for AGI tasks, nor did it explicitly set out to be a testbed for the sort of tasks prescribed by Psychometric AI.

3.2 The Architecture of a PAGI World Setup

Figure 2 pictures the architecture of a typical PAGI World + AI controller pairing. As the figure illustrates, it is helpful to think of the processes controlled by the PAGI-World application to be the PAGI side, as opposed to the side which can be completely implemented externally, referred to as the AI side. The reflex and state machine described in Section 3.3 is controlled and managed on the PAGI side, but both states and reflexes can be dynamically modified through commands sent by the AI side.

All commands going from the AI side to the PAGI side, and all sensory information passing in the other direction, is passed via messages communicated through TCP/IP ports. Therefore, the AI-side can be written in any programming language that supports the creation, and decoding, of strings over TCP/IP. Although this flexibility sets PAGI World apart from many other alternatives, some may prefer an additional level of abstraction on the AI side, and for this reason we provide, and are continuing development on, a Python API called *pyPAGI*.

Tasks can be created, saved, and loaded using the GUI editor at run-time (Figure 1), but as suggested by Figure 2, they can also be somewhat configured by AI-side commands. This can be useful to modify the layout of the task dynamically in response to actions the AI agent takes (e.g., making an apple appear as a reward, or a bottle of poison as a punishment), or to load new tasks after successful task completion for automated batch processing of tasks.

3.3 Reflexes, DFAs, and the Implicit vs. Explicit Distinction

Although communication through TCP/IP ports is relatively quick, and the command system we have created is designed to be efficient, there are some actions that require extremely rapid, simple checks and responses. For example, holding an object in the air at a certain position relative to the body for an extended period of time may require many quick corrections. If the object starts to move down, more upward force should be applied. But if it moves too far up, downward force should be applied (or the amount of upward force should be reduced). In order to hold the object as still as possible, the amount of force applied would be based on its current and projected velocity and position. However, if the AI script requests this information, does a calculation to determine the amount of correction required, and sends back the command to adjust the amount of force, by the time this command is received by PAGI World and processed it may be inaccurate.

PAGI World fixes this problem by implementing *states* and *reflexes*. Reflexes and states can be set and modified through commands from the AI script, but they are actually checked and executed completely on the PAGI-World side, which allows for much faster reaction times. A reflex r consists of a tuple $(\mathbf{C}, \mathbf{A})$, where $\mathbf{C}$ is a list of conditions and $\mathbf{A}$ is a list of actions. Each condition in $\mathbf{C}$ must be satisfied in order for reflex r to activate. These conditions can consist of sensory inequalities, for example: whether one of the tactile sensors detects a temperature above a certain amount, or whether the AI agent's body is moving above a certain velocity. If all of the conditions are met, then the actions are executed immediately. Furthermore, sensory inequalities can be specified as simple arithmetic functions of sensory values, so that a reflex can be fired if (to cite an arbitrary example) the horizontal component of the agent's body's velocity is at least twice the value of the vertical component of its velocity.

States can be activated and checked by reflexes. Essentially, this means that multiple deterministic finite automata (DFAs) can be stored and executed completely on the PAGI side. However, the expressivity of the conditions and actions within each reflex strictly restricts the system so that full Turing machines cannot be implemented on the PAGI side. This allows developers to implement two important categories of abilities generally regarded to be part of the human experience: explicit, and implicit. Recall that the explicit vs. implicit distinction divides the mind into explicit processes which are generally slow, deliberate, and easy to verbalize, versus implicit processes which are mostly quick, automatic, and not easily accessible to the conscious mind [35].

The implicit/explicit distinction [35], which roughly parallels the System 1/System 2 distinction of Kahneman [20] (but see [37] for a criticism of System 1 vs. 2), encompasses an extremely broad spectrum of explanations for human phenomena [34, 35, 36]. If a simulation environment restricts itself to AI controllers that rely on explicit or implicit processes exclusively, then it cannot hope to capture the breadth of tasks required to qualify a Psychometric Artificial *General* Intelligence. If PAGI tasks are meant to subsume all tasks solvable by neurobiologically normal human adults, then a simulation environment designed to capture PAGI tasks should also be able to test AI agents on their use of both explicit and implicit knowledge.

Although the PAGI side does not support all imaginable implicit processes (for example, some might believe that a Bayesian probabilistic approach or a Deep Neural Network is necessary to implement some implicit processes), the fact that multiple DFAs can be stored and executed in PAGI World's optimized code gives the user flexibility to capture a wide range of implicit processes. Furthermore, in keeping with the design principles of PAGI World, AI systems built on implicit processes can still be implemented fully on the AI side.

4 Some Example Tasks

We have designed some tasks to demonstrate the range of possibilities and showcase some of PAGI World's unique features.

4.1 Piaget-MacGyver Rooms

4.1.1 The Water Diversion Piaget-MacGyver Room

A prime example of a typical MacGyver task comes from Season 2, Episode 5 of the MacGyver television series. Angus MacGyver, the series' titular character, found a friend of his being threatened by a mountain lion. MacGyver, positioned at a ledge above both his friend and the mountain lion, reconfigured some rocks and a log so that he could guide a nearby stream of water in such a way that it created a small waterfall separating his friend and the mountain lion. The mountain lion ran away immediately.

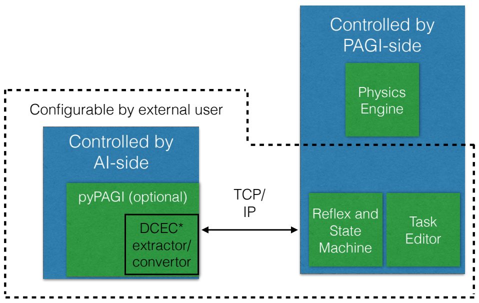


Figure 2: The architecture of an instance of PAGI World and an AI controller. Everything on the AI side can be written by AI researchers, as the interface with the PAGI side is handled through messages passed over TCP/IP sockets. A Python library, called *pyPAGI*, is also optionally available to assist researchers with common AI-side functionality, including encoding of PAGI-World knowledge in the Deontic Cognitive Event Calculus ($\mathcal{DCEC}^*$). The reflex/state machine and task editors can also be controlled through TCP/IP, though the task editor is additionally available through a WYSIWYG drag-and-drop interface.

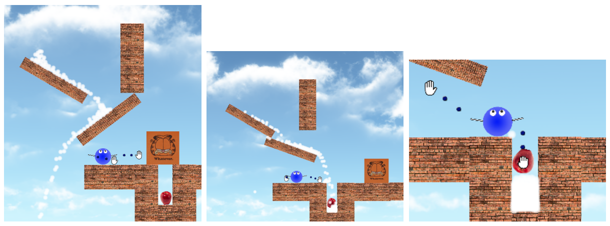


Figure 3: A sample Piaget-MacGyver Room in which the agent is expected to direct the flow of water in order to reach an apple.

Of course, other solutions may have been available. Perhaps MacGyver could have simply thrown rocks at the mountain lion, or fashioned a bow and arrow out of twigs, sharpened stones, and parts of his knapsack. But these different solutions would have come with their own unique advantages and disadvantages, and furthermore, to not lose sight of the PAGI-oriented question: Could an artificially intelligent agent figure out *any* of these solutions without having been specifically trained for that particular solution? PAGI problems such as the Piaget-MacGyver room challenge researchers to find answers to this question.

The ability to direct the flow of water opens up a wide range of tasks, which we can model in PAGI World. Using the Fluvio library for fluid dynamics, PAGI World can generate fluid-like particles. These particles have several realistic properties of fluids; for example, when poured into a cup-like container, an object placed in the filled container will either float or sink depending on its weight. The task in Figure 3 has water flowing down a system of angled brick walls. The bottom angled wall can be moved around a pivot in its center, so that the flow of water can be directed to the left or right. If it is directed to the right, the water will flow further down until it encounters an orange block. If the orange block is moved, the water will eventually flow into a pit which contains an apple, which the agent could previously see but not reach. The flow of water will eventually fill up the pit, causing the apple to float up to where the agent can reach it.

4.1.2 The Piagetian Balance-Beam Task

Some tasks that might be considered Piaget-MacGyver Rooms can come directly from classical Piagetian experiments. Inhelder and Piaget's (1958) Balance-Beam Task (BBT) [19] has been modeled many times using a variety of modeling techniques [41, 30, 32, 31, 38, 23]. In the BBT, a balancing beam with a set of weights are provided to a subject. The balancing beam has notches, hooks, or some other apparatus that allows the weights to be placed on the left or right sides of the balancing beam at predefined intervals. In most versions of the task, the values of the weights and the distances that the locations are from the center are made available to the subject. The task is normally to figure out some version of the torque rule, which relates the product of the value of a weight and its distance from the center. For example, the subject may be presented with a configuration of weights on the scale, and the subject is asked to predict whether the right or left side will tilt downwards or the scale will balance.

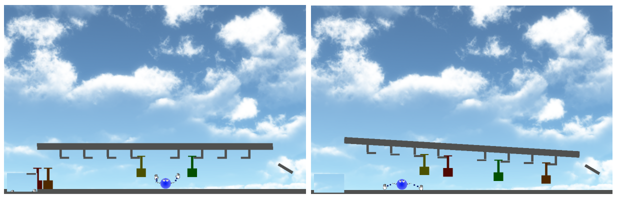


Figure 4: The Piagetian Balance Beam Task in PAGI World

We recreate the BBT in PAGI World as in Figure 4. A balance beam with several hooks on which weights can be hung is in the center of the screen. There is also a switch (on the bottom right side of the screen) that can be used to toggle the motion of the balance beam. That way, the weights can be hung on the balance beam and the beam will not move while the agent is reasoning about how the beam should tilt when the switch is toggled. This BBT is fully implemented and included in PAGI World (although no AI capable of solving this task has yet been developed).

One clear limitation of computationally modeling most Piagetian tasks is that you can't really communicate with the AI agent in natural language like you can with the children in Piagetian experiments. Although the current state of the art in natural-language processing and generation prohibits such communication at present, PAGI World offers tools to make it easier for researchers who are trying to achieve this benchmark. There is a way to "talk to" the AI agent through an input text box in PAGI World itself. Having the agent talk back, however, can be handled in three possible ways: through simple output handled completely by the code on the AI side, by sending a message to PAGI World that can be displayed in an output window (a console screen accessible through PAGI World), or by creating a speech bubble.

Speech bubbles can be created (Figure 5) by AI-side scripts. These speech bubbles are recognizable by PAGI guy's vision system, along with data such as the name of the speaker, the location of the box, the text written inside of it, and so on.

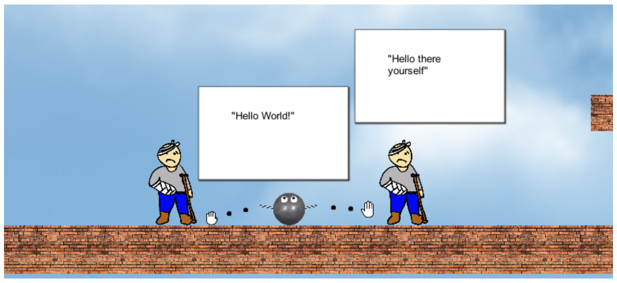


Figure 5: Speech bubbles can be created to simulate conversations, giving them a visual element that makes for understandable demonstrations.

4.2 Moral Reasoning

4.2.1 *Resolving a Moral Dilemma*

One interesting and possibly fertile source of PAGI-World tasks is the area of morality, specifically machine ethics [40], which is devoted to trying to engineer machines that have at least a degree of moral sensibility. Figure 6 depicts an example task in which two injured soldiers are equally distant from PAGI guy, and only a single health kit is available. PAGI World allows the visual sensors to detect whether a soldier is injured or healthy. If a health kit touches an injured soldier, the soldier will become healthy and the health kit will disappear.

Some might recognize this task as a variant of the *Buridan's Donkey* scenario, where a donkey unable to choose between two bales of hay, paralyzed by indecision, ends up starving to death. This task can be used, for example, to evaluate whether a reasoning system is sufficiently intelligent to avoid certain paralyzing situations, or to search for creative solutions where they exist. PAGI World provides two health kits: a small one, and a large one. The large health pack (as in Figure 6) can be broken in half, when enough force is applied to it, and each half can be given to one of the soldiers. We recently used this task to demonstrate that the search for creative solutions, and the ultimate decision to let one soldier remain injured if such a solution can not be found, can be carried out entirely within the framework provided by the Deontic Cognitive Event Calculus [5].

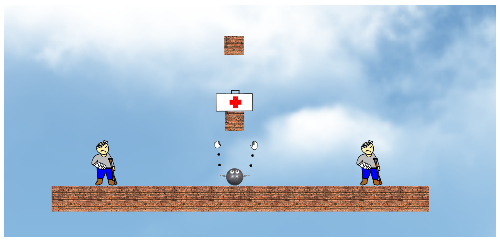


Figure 6: A task where an agent only has one health kit but two injured soldiers.

4.2.2 *Reasoning Over Obligations to Detect Hijacking*

Bello et al. in [1], use PAGI World to model a situation in which a robot capable of autonomously reasoning over moral obligations (represented by PAGI guy) is maliciously attacked and infected with a virus. The virus hijacks the robot's actuators, instructing them to perform the action of pushing a healthy soldier over a ledge, resulting into the soldier's falling into a lava pit below. By using the reasoning mechanisms provided by the Deontic Cognitive Event Calculus, Bello et al. are able to show that the robot can reason over its obligations and its apparent desire to perform the harmful action to the soldier, ultimately concluding that such an action is not morally permissible. Presumably, further reasoning might hypothesize the existence of the hijacking virus.

4.3 Low-Level Reasoning

4.3.1 *Learning to Catch and to Avoid Painful Objects*

The reflex and state system (Section 3.3) allows for *reflexes* to be created and configured from the AI side and stored on the PAGI side, where they can be activated automatically based on sensory values. Whenever such a reflex is activated, a message can optionally be sent to the AI side, naming the reflex which was just fired. This can allow for after-the-fact reasoning of automatic reflexes. In one example, a set of reflexes are configured so that the agent will quickly retract its hand after touching a hot object (temperature sensors are among the sensors lining the circumference of both hands). If the agent touches an object with a high temperature (the flames and the hot plates, in this example), the reflex will first fire and then the AI side can decide if further actions should be taken, for example by looking at the object which caused the reflex to fire, thus giving the agent a motivation to avoid going near hot objects in the future.

PAGI World also allows for objects in the simulation to have an objective utility value. These are called *endorphins*; objects with a positive endorphin value are those that the AI agent may want to pursue (e.g. food items), whereas negative endorphin values are those the agent should avoid. Most objects in PAGI World have an endorphin value of zero, and PAGI World itself does not ensure that certain endorphin-seeking behaviors are implemented by the agent.

Another example of the reflex system is catching an object. In the baseball-catching task (available online), a baseball is launched to soar right above the agent's head, and he has a time period of a little over a second in which his hands can reach the ball without moving his body. Of course, the agent can optionally move his body as well to reach for the ball if necessary. If the ball is caught, the agent will receive an endorphin bonus; otherwise, the ball will fall off of the ledge and no longer be accessible to the agent. Although TCP/IP communication is relatively quick, the microsecond timing required to estimate the speed and trajectory of the ball and move the hand in time to catch it is not easily (and probably not possibly) done with controls firmly on the AI side. Instead, reflexes must be used.

In addition to sensor values, reflexes can be configured to activate, deactivate, or check for *states*, which are simply string labels configured by the AI side. Because reflexes can optionally be set to fire only if certain states are active, followed by deactivation of current states and activation of new ones, reflexes can be used as transitions between states in a deterministic finite-automata machine. In addition to receiving notifications whenever such a reflex fires, the AI-side can poll the PAGI side to see which states and reflexes are currently active, so that it can change them if necessary.

4.4 High-Level Reasoning

4.4.1 *Self-Awareness: Reasoning About the Self*

Whether artificial agents can ever have self-awareness is a highly controversial topic; this is clear from the public discussion surrounding popular press reporting a recent experiment in robot self-awareness. Bringsjord et al. (2015) started with a puzzle devised by philosopher Luciano Floridi [15], in which three artificial agents, or robots, are given either a "dumbing" pill or a placebo. The dumbing pill disables their higher-level cognition (i.e. their ability to reason), and is given to two of the three robots. All three robots are then asked which pill they received. The two given the dumbing pill cannot reason, and thus remain silent (note their silence is not by choice, but because they fail to reason at all). The robot given the placebo ultimately concludes it cannot decide, whereupon it utters the phrase "I don't know." However, upon hearing (and, importantly, feeling) itself utter the phrase, it now has a new piece of knowledge with which to reason. Given this new piece of knowledge, along with an understanding of the rules of the current experiment (knowledge which is also initially given to the other two robots), it can then conclude that it was in fact, not given the dumbing pill.

PAGI World was used to create Floridi's task [8]. We created a task containing three agents[5], and on the AI-side, the agents were connected to an automated theorem prover reasoning in the Deontic Cognitive Event Calculus [5]. This calculus is a knowledge representation framework capable of expressing *de se* beliefs, i.e. a specific type of reasoning about the self [4]. Pills are given to each agent, implemented as endorphin-producing items. Placebos produce negative endorphins, and the dumbing pills produce positive endorphins. When a pill is absorbed by an agent, the endorphin value is sent to the AI-side, where the agent's reasoning system is either disabled or left untouched.

The experiment can then proceed. Commands "spoken" to the agents are typed in to a text box that PAGI World makes available (strings entered into this box are sent as messages to the AI-side). Statements the agents wish to output can be sent from the AI-side to PAGI-side as a special type of command, where they will be displayed on a debug screen.

It is important to here introduce a disclaimer: we do not claim such an experiment "proves" self-awareness in our artificial agent. Rather, this is just another example of the psychometric philosophy underlying PAGI World, according to which a series of psychometric tests are proposed (in this case, the tests given by Floridi (2005)), undertaken by AI researchers, and the cycle repeats. PAGI World makes the creation of such tests easier.

4.4.2 Analogico-Deductive Reasoning

Because tasks can be created dynamically through special commands sent from the AI-side, saved to file, and loaded, it is possible to train PAGI guy on tasks, then change the task and see how the previous training transfers. Such a setup is ideal in the testing of transfer learning, e.g. in instances of analogico-deductive reasoning (ADR). In ADR, an agent uses analogical reasoning to generate a hypothesis about some target domain (where the target domain is often unfamiliar in some way). The hypothesis is then subjected to deductive reasoning, so that it can either be proven false, or possibly shown to imply some method of experimental verification [23].

In [24], Marton et al. used PAGI World to demonstrate ADR in a real-time task. The agent was given the ability to assess its surroundings, and to solve a source task. In the source task, PAGI guy had to reach an apple by jumping over an obstacle (a raised brick). It was then faced with a different task, in which the obstacle was instead a gap in the ground. The AI-side was able to generate a hypothesis that the solution used in the source task, to jump over the obstacle, would work in the target task as well. It then carries out a proof using a deductive reasoner, and upon receiving confirmation from the prover, jumps over the gap to reach the apple. For more details, see [24].

[5] The ability to create multiple agents in PAGI World is a beta feature that is not available in the publicly available version. We expect this feature to be ready soon.

5 Conclusion and Future Work

The release of PAGI World is accompanied with a call to all AGI and human-level-AI researchers to finally examine the strengths and limits of their preferred approaches. PAGI World allows for researchers to very easily create tasks and microworlds in a 2D world with realistic physics, with no knowledge in how to program. PAGI World can interact with AI agents that are written in virtually any programming language, and the simulation can be run on any major operating system. We have very carefully designed PAGI World to have an extremely low technical barrier, so that many researchers can find common ground upon which to compare their different approaches.

PAGI World can also be used in educational applications. In the Spring of 2015, the present authors taught a course at RPI, which made use of PAGI World for homework assignments, projects, and to give students hands-on experience in implementing AI and cognitive modeling techniques. The experiment was quite a success, and we are currently further exploring PAGI World's possibilities for education.

The future of PAGI World is bright. We already have several AI systems in progress whose goals are to solve already-finished PAGI World tasks, and as development continues we hope to greatly increase the number of tasks which are available and the sophistication of the agents which solve those tasks. The library of future tasks, we hope, will diversify and reflect the broad spectrum of tasks which require human-like intelligence.

PAGI World downloads, documentation, example tasks, and source code can be downloaded from the RAIR Lab website at `http://rair.cogsci.rpi.edu/projects/pagi-world`.[6]

REFERENCES

[1] Paul Bello, John Licato, and Selmer Bringsjord, 'Constraints on Freely Chosen Action for Moral Robots: Consciousness and Control', in *Proceedings of RO-MAN 2015*, (2015).

[2] Mark H. Bickhard, 'The Interactivist Model', *Synthese*, **166**(3), 547–591, (July 2008).

[3] Selmer Bringsjord, 'Psychometric Artificial Intelligence', *Journal of Experimental and Theoretical Artificial Intelligence*, **23**(3), 271–277, (2011).

[4] Selmer Bringsjord and Naveen S. Govindarajulu, 'Toward a Modern Geography of Minds, Machines, and Math', *Philosophy and Theory of Artificial Intelligence*, **5**, 151–165, (2013).

[5] Selmer Bringsjord, Naveen S. Govindarajulu, Simon Ellis, Evan McCarty, and John Licato, 'Nuclear Deterrence and the Logic of Deliberative Mindreading', *Cognitive Systems Research*, **28**, 20–43, (2014).

[6] Selmer Bringsjord and John Licato, 'Psychometric Artificial General Intelligence: The Piaget-MacGyver Room', in *Theoretical Foundations of Artificial General Intelligence*, eds., Pei Wang and Ben Goertzel, 25–48, Atlantis Press, 8, square des Bouleaux, 75019 Paris, France, (2012).

[7] Selmer Bringsjord, John Licato, and Alexander Bringsjord, 'The Contemporary Craft of Creating Characters Meets Today's Cognitive Architectures: A Case Study in Expressivity', in *Integrating Cognitive Architectures into Virtual Character Design*, eds., Jeremy Turner, Michael

[6] Special thanks to a series of brilliant undergraduate students who contributed to earlier versions of PAGI World: Jack Cusick, Robert Dong, Kainoa Eastlack, Nick Marton, and Spencer Whitehead. This work was supported in part by ONR grant N000141310342, and Licato's AFOSR Young Investigator Research Program grant.

Nixon, Ulysses Bernardet, and Steve DiPaola, Advances in Computational Intelligence and Robotics (ACIR), Information Science Reference / IGI Global, (Forthcoming).

[8] Selmer Bringsjord, John Licato, Naveen S. Govindarajulu, Rikhiya Ghosh, and Atriya Sen, 'Real Robots that Pass Human Tests of Self-Consciousness', in *Proceedings of RO-MAN 2015*, (2015).

[9] Selmer Bringsjord and Bettina Schimanski, 'What is Artificial Intelligence? Psychometric AI as an Answer', in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, (2003).

[10] Rodney A. Brooks, 'Elephants Don't Play Chess', *Robotics and Autonomous Systems*, **6**, 3–15, (1990).

[11] Rodney A. Brooks, 'Intelligence Without Representation', *Artificial Intelligence*, **47**, 139–159, (1991).

[12] David Bruce, 'Cognitive Architecture and Testbed for a Developmental AI System'. Undergraduate Honours Thesis, 2010.

[13] Harold H. Chaput, Benjamin Kuipers, and Risto Miikkulainen, 'Constructivist Learning: A Neural Implementation of the Schema Mechanism', in *Proceedings of the Workshop on Self Organizing Maps (WSOM03)*. FCFM, (2003).

[14] Gary L. Drescher, *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*, The MIT Press, 1991.

[15] Luciano Floridi, 'Consciousness, Agents and the Knowledge Game', *Minds and Machines*, **15**(3-4), 415–444, (2005).

[16] Dedre Gentner and Kenneth Forbus, 'Computational Models of Analogy', *Wiley Interdisciplinary Reviews: Cognitive Science*, **2**(3), 266–276, (2011).

[17] Frank Guerin, 'Learning Like a Baby: A Survey of Artificial Intelligence Approaches', *The Knowledge Engineering Review*, **26**(2), 209–236, (2011).

[18] S. Harnad, 'The Symbol Grounding Problem', *Physica D*, **42**, 335–346, (1990).

[19] Barbel Inhelder and Jean Piaget, *The Growth of Logical Thinking: From Childhood to Adolescence*, Basic Books, Inc, 1958.

[20] Daniel Kahneman, *Thinking, Fast and Slow*, Farrar, Straus and Girous, 2011.

[21] John Licato, *Analogical Constructivism: The Emergence of Reasoning Through Analogy and Action Schemas*, Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, NY, May 2015.

[22] John Licato, Selmer Bringsjord, and Naveen S. Govindarajulu, 'How Models of Creativity and Analogy Need to Answer the Tailorability Concern', in *Proceedings of the IJCAI 2013 Workshop on Computational Creativity, Concept Invention, and General Intelligence*, eds., Tarek Richard Besold, Kai-uwe Kühnberger, Marco Schorlemmer, and Alan Smaill, Beijing, China, (2013).

[23] John Licato, Selmer Bringsjord, and John E. Hummel, 'Exploring the Role of Analogico-Deductive Reasoning in the Balance-Beam Task', in *Rethinking Cognitive Development: Proceedings of the 42nd Annual Meeting of the Jean Piaget Society*, Toronto, Canada, (2012).

[24] Nick Marton, John Licato, and Selmer Bringsjord, 'Creating and Reasoning Over Scene Descriptions in a Physically Realistic Simulation', in *Proceedings of the 2015 Spring Simulation Multi-Conference*, (2015).

[25] Jean Piaget and Rolando Garcia, *Understanding Causality*, W.W. Norton and Company, Inc., New York, New York, USA, 1974.

[26] Jean Piaget and Barbel Inhelder, *The Growth of Logical Thinking*, Basic Books, Inc, New York, New York, USA, 1958.

[27] Jean Piaget, Jacques Montangero, and J.B. Billeter, *Studies in Reflecting Abstraction*, Psychology Press, 1976/2001.

[28] Jean Piaget, Jacques Montangero, and J.B. Billeter, 'The Formation of Analogies', in *Studies in Reflecting Abstraction*, ed., Robert Campbell, Psychology Press, (2001).

[29] Steven R Quartz and Terrence J. Sejnowski, 'The Neural Basis of Cognitive Development: A Constructivist Manifesto', *Behavioral and Brain Sciences*, **20**, 537–596, (1997).

[30] S. Sage and Pat Langley, 'Modeling Cognitive Development on the Balance Scale Task', in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, ed., A. Bundy, volume 1, pp. 94–96, (1983).

[31] William C. Schmidt and C. X. Ling, 'A Decision-Tree Model of Balance Scale Development', *Machine Learning*, **24**, 203–230, (1996).

[32] Thomas R. Shultz, Denis Mareschal, and William C. Schmidt, 'Modeling Cognitive Development on Balance Scale Phenomena', *Machine Learning*, **16**, 57–86, (1994).

[33] Peter Smith, *An Introduction to Gödel's Theorems*, Cambridge University Press, Cambridge, UK, 2007.

[34] Ron Sun, 'From Implicit Skills to Explicit Knowledge: A Bottom-Up Model of Skill Learning', *Cognitive Science*, **25**(2), 203–244, (2001).

[35] Ron Sun, *Duality of the Mind: A Bottom Up Approach Toward Cognition*, Lawrence Erlbaum Associates, Mahwah, NJ, 2002.

[36] Ron Sun. Desiderata for Cognitive Architectures. Website, September 2003.

[37] Ron Sun, 'Interpreting Psychological Notions: A Dual-Process Computational Theory', *Journal of Applied Research in Memory and Cognition*, **In Press**, (2014).

[38] Geoffrey F. W. Turner and Hoben Thomas, 'Bridging the Gap between Theory and Model: A Reflection on the Balance Scale Task', *Journal of Experimental Child Psychology*, **81**, 466–481, (2002).

[39] Ernst von Glasersfeld, 'Abstraction, Re-Presentation, and Reflection: An Interpretation of Experience and of Piaget's Approach', in *Epistemological Foundations of Mathematical Experience*, ed., L. P. Steffe, 45–67, Springer, New York, New York, USA, (1991).

[40] Wendell Wallach and Colin Allen, *Moral Machines: Teaching Robots Right From Wrong*, Oxford University Press, Oxford, UK, 2008.

[41] Friedrich Wilkening and Norman H. Anderson, 'Comparison of Two Rule-Assessment Methodologies for Studying Cognitive Development and Knowledge Structure', *Psychological Bulletin*, **92**(1), 215–237, (1982).

:

Historical account of computer models solving IQ test problems

Fernando Martínez-Plumed[1] and **José Hernández-Orallo**[2] and **Ute Schmid**[3] and **Michael Siebers**[4] and **David L. Dowe**[5]

Abstract. In this short paper we summarise our work in [9] where we make a review of what has been done when intelligence test problems have been analysed through cognitive models or particular systems. This work has been motivated by an observed explosion of the number of papers on this topic in recent years. We have made a general account of all these works in terms of how they relate to each other and what their real achievements are. Not only do we aim at analysing the meaning, utility, and impact of these computer models, but also better understanding what these tests measure in machines, whether they are useful to evaluate AI systems, whether they are really challenging problems, and whether they are useful to understand (human) intelligence.

1 INTRODUCTION

In the early days of artificial intelligence, the IQ test classical approach to human intelligence evaluation was considered useful not only as a tool for the study of cognitive processes and the development of new techniques, but also for the evaluation of AI systems or even as the goal for AI research. Since then, human psychometric tests have been repeatedly suggested as a much better alternative to most task-oriented evaluation approaches in AI. The question thus is whether this measurement of mental developmental capabilities leads to a feasible, practical evaluation for AI.

In this paper we briefly review our work in [9], where we analysed all the computer models taking intelligence tests (or as many as we could find, about thirty in total), starting with Evans's ANALOGY [6] and going through to Spaun [5], a noteworthy 2.5-million-neuron artificial model brain. This analysis was motivated by an observed explosion in recent years of the number of papers featuring computer models addressing intelligence test problems. featuring computer models addressing intelligence tests problems. We wanted to investigate whether this increase was casual or was motivated by an increasing need of these tests and the computer models solving them. Overall, the main goal of the paper was to understand the meaning, utility, and impact of these computer models taking intelligence tests, and explore the progress and implications of this area of research.

2 HISTORICAL ACCOUNT

The relation between artificial intelligence and psychometrics started more than fifty years ago. As early as 1963, Evans [6] and Simon and Kotovsky [18] devised AI programs able to identify regularities in patterns (respectively, analogy tasks and letter series completion problems).

After the initial interest of AI research in IQ test problems, this branch of research sank into oblivion during the twenty of so years. However, since the 1980s, cognitive science research recovered this line of research. Hofstadter developed a series of computational models in the Copycat project [11] with the major goal of understanding analogy. In the 1990s, some cognitive models were proposed to simulate the human cognitive processes that take place when solving inductive inference IQ test problems [2].

In AI, forty years after the work of Evans and Simon & Kotovsky, in 2003, computer programs solving intelligence tests became of interest again. On one hand, Sanghi and Dowe [16] wanted to make a conclusive point about how easy it was to make non-intelligent machines pass intelligence tests. This could have dealt a definitive deathblow to this already ebbing approach. On the other hand, Bringsjord and Schimanski aimed at resuscitating the role of psychometric tests—including not only intelligence tests but also tests about personality, artistic creativity, etc.—in AI [1]. They claimed that psychometric tests should not be dismissed but placed at a definitional, major role for what artificial intelligence is and proposed "psychometric artificial intelligence" (PAI) as a direction of research.

But the fact is that the past ten (and especially five) years (since 2006 and especially 2011) have seen a boom of computational models aimed at solving intelligence test problems. The diversity of goals and approaches has also widened, including the use of intelligence tests for the analysis of what intelligence is, for the understanding of certain aspects of human cognition, for the evaluation of some AI techniques or systems, including robots, and, simply, to have more insights about what intelligence tests really represent. See [9, Section 5] for what we hope has been a complete description of all the computer models that have addressed intelligence tests and related tests, presented in chronological order.

3 DISCUSSION

The analysis has not been restricted to performing a survey of all models addressing intelligence tests. Through a comprehensive account of the models we derived a set of criteria aiming at understanding the meaning, utility, and impact of these computer models taking intelligence tests, and explore the progress and implications of this area of research. Furthermore, this analysis helped us to have a better understanding of the relevance and (the limited) connections of these approaches, and draw some conclusions about their usefulness.

We have seen that most approaches are very recent [5, 12, 14, 17, 20, 21, 15, 10]. Is it an indication of relevance? According to the

[1] Universitat Politècnica de València, Spain, email: fmartinez@dsic.upv.es
[2] Universitat Politècnica de València, Spain, email: jorallo@dsic.upv.es
[3] University of Bamberg, Germany, email: ute.schmid@uni-bamberg.de
[4] University of Bamberg, Germany, email: michael.siebers@uni-bamberg.de
[5] Monash University, Australia, email: david.dowe@monash.edu

publication venues, we have seen that they go from mainstream AI to cognitive science, or even psychology, and some of them are in leading conferences and journals in these areas or even in interdisciplinary general outlets. However, it seems that most approaches aim at unveiling general (artificial) intelligence principles in ways that are not necessarily connected to the way humans solve these tests. In fact, there is a wide variety in the techniques used, from more adhoc to more general AI techniques (mostly from machine learning, pattern recognition, automated reasoning, and natural language processing). This suggests that this is attracting more interest in artificial intelligence and cognitive science than in psychology. Overall, some of these models (anthropomorphic or not) have been useful to provide insights and valuable information about how human cognition works.

What about the use of these tests for AI evaluation? Are they becoming more common? It has been recently argued—from human intelligence researchers—that intelligence tests are the right tool to evaluate AI systems [3]. Nonetheless, we have not seen that artificial intelligence has changed its evaluation protocols following this increase of models taking intelligence tests (witch a few exceptions such as [7, 19, 17, 5]). Furthermore, we have seen that even for supposedly general tasks that are designed for evaluation, many approaches have the (understandable) tendency to specialise to the task and hard-wire parts (or most) of the solution. The key issue is thus to consider a greater diversity of problems. Very few approaches address more than one kind of test. Actually, the more specific a test is the easier it is to develop specific solutions. Furthermore, different presentations and difficulty levels should be explored. The categories and overlaps between problems could be assessed via theoretical models, instead of using factor analysis as in psychometrics. In other words, a theoretical alternative to the classification of mental abilities should be considered (see [8, 4]).

There is also a huge diversity in whether performance and difficulty are assessed. We need to be clear that focussing on the overall results of a computer model and comparing them with the results of humans is not very informative about how challenging the problem is. Humans are general-purpose systems and it is not fair to compare them with some systems that are only able to solve one problem—even if the problem comes from an intelligence test. Furthermore, many of these intelligence test problems have been developed for humans, and hence it can be unfair to evaluate AI systems' limitations with anthropocentric measures. Nonetheless, some of the works perform an interesting analysis in terms of difficulty. The purpose is to determine what instances are more difficult, but this is not very related to how challenging the problem is. In fact, focussing on the most difficult problems may even make the system more specialised to the intelligence test task at hand. Some of the previous works have studied whether difficulty is related to the size of the working memory, the size of the pattern, the number of elements that need to be combined or retrieved from background knowledge or the operational constructs needed to solve this problems [18, 2, 20, 21, 13]. These notions of difficulty are much more general and can work independently of the problem and the representation.

4 CONCLUSION

Of the approximately 30 papers we have analysed in [9], half of them have appeared in the past five years. We wanted to investigate whether this increase was casual or was motivated by an increasing need of these tests and the computer models solving them. In order to study this we soon realised that computer models addressing intelligence tests may have different purposes and applications: to advance AI by the use of challenging problems (this is the Psychometric AI approach), to use them for the evaluation of AI systems, to better understand intelligence tests and what they measure (including item difficulty) and, finally, to better understand what (human) intelligence is. Furthermore, the use of intelligence tests for AI evaluation has provided very insightful information about what intelligence tests measure and what they do not and, ultimately, about what characterises intelligence in humans.

REFERENCES

[1] S. Bringsjord and B. Schimanski, 'What is artificial intelligence? Psychometric AI as an answer', in *International Joint Conference on Artificial Intelligence*, pp. 887–893, (2003).

[2] P. A. Carpenter, M. A. Just, and P. Shell, 'What one intelligence test measures: A theoretical account of the processing in the Raven Progressive Matrices test', *Psychological review*, **97**, 404–431, (1990).

[3] D. K. Detterman, 'A challenge to Watson', *Intelligence*, **39**(2-3), 77–78, (2011).

[4] D.L. Dowe and J. Hernández-Orallo, 'How universal can an intelligence test be?', *Adaptive Behavior*, **22**(1), 51–69, (2014).

[5] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, C. Tang, and D. Rasmussen, 'A large-scale model of the functioning brain', *Science*, **338**(6111), 1202–1205, (2012).

[6] T. Evans, 'A heuristic program to solve geometric-analogy problems', in *Proc. SJCC*, volume 25, pp. 327–339, (1965). vol. 25.

[7] S. Griffith, J. Sinapov, V. Sukhoy, and A. Stoytchev, 'A behavior-grounded approach to forming object categories: Separating containers from noncontainers', *Autonomous Mental Development, IEEE Transactions on*, **4**(1), 54–69, (2012).

[8] J. Hernández-Orallo and D. L. Dowe, 'Measuring universal intelligence: Towards an anytime intelligence test', *Artificial Intelligence*, **174**(18), 1508–1539, (2010).

[9] J. Hernández-Orallo, F. Martínez-Plumed, Ute Schmid, Michael Siebers, and David L. Dowe, 'Computer models solving intelligence test problems: Progress and implications', *Artificial Intelligence*, **230**, 74 – 107, (2016).

[10] J. Hofmann, E. Kitzelmann, and U. Schmid, 'Applying inductive program synthesis to induction of number series a case study with IGOR2', in *KI 2014: Advances in AI*, pp. 25–36. Springer, (2014).

[11] D. R. Hofstadter and M Mitchell. The copycat project, 1984.

[12] A. Lovett and K. Forbus, 'Modeling multiple strategies for solving geometric analogy problems', in *Proceedings of the 34th Annual Conference of the Cognitive Science Society*, pp. 701–706, (2012).

[13] F. Martínez-Plumed, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana, 'A computational analysis of general intelligence tests for evaluating cognitive development', *submitted (second revision)*, (2016).

[14] H. Prade and G. Richard, 'From analogical proportion to logical proportions: A survey', in *Computational Approaches to Analogical Reasoning: Current Trends*, volume 548 of *Studies in Computational Intelligence*, 217–244, Springer Berlin Heidelberg, (2014).

[15] M. Ragni and S. Neubert, 'Analyzing ravens intelligence test: Cognitive model, demand, and complexity', in *Computational Approaches to Analogical Reasoning: Current Trends*, volume 548 of *Studies in Computational Intelligence*, 351–370, Springer, (2014).

[16] P. Sanghi and D. L. Dowe, 'A computer program capable of passing IQ tests', in *4th Intl. Conf. on Cognitive Science (ICCS'03), Sydney*, pp. 570–575, (2003).

[17] C. Schenck, *Intelligence tests for robots: Solving perceptual reasoning tasks with a humanoid robot*, Master's thesis, Iowa State Univ., 2013.

[18] H. A. Simon and K. Kotovsky, 'Human acquisition of concepts for sequential patterns.', *Psychological Review*, **70**(6), 534, (1963).

[19] J. Sinapov and A. Stoytchev, 'The odd one out task: Toward an intelligence test for robots', in *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pp. 126–131. IEEE, (2010).

[20] C. Strannegård, M. Amirghasemi, and S. Ulfsbücker, 'An anthropomorphic method for number sequence problems', *Cognitive Systems Research*, **2223**, 27–34, (2013).

[21] C. Strannegård, A. Nizamani, A. Sjöberg, and F Engström, 'Bounded Kolmogorov complexity based on cognitive models', in *Artificial General Intelligence*, eds., K. U. Kühnberger, S. Rudolph, and P. Wang, volume 7999 of *LNCS*, 130–139, Springer Berlin Heidelberg, (2013).

Expert and Corpus-Based Evaluation of a 3-Space Model of Conceptual Blending

Donny Hurley and **Yalemisew Abgaz** and **Hager Ali** and **Diarmuid O'Donoghue**[1]

Abstract. This paper presents the 3-space model of conceptual blending that estimates the figurative similarity between Input spaces 1 and 2 using both their analogical similarity and the inter-connecting Generic Space. We describe how our Dr Inventor model is being evaluated as a model of lexically based figurative similarity. We describe distinct but related evaluation tasks focused on 1) identifying novel and quality analogies between computer graphics publications 2) evaluation of machine generated translations of text documents 3) evaluation of documents in a plagiarism corpus. Our results show that Dr Inventor is capable of generating novel comparisons between publications but also appears to be a useful tool for evaluating machine translation systems and for detecting and assessing the level of plagiarism between documents. We also outline another more recent evaluation, using a corpus of patent applications.

Introduction

Analogical reasoning and conceptual blending have been identified by cognitive science as central abilities of human intelligence. Their relevance to general (artificial) intelligence being highlighted by their role in process like: learning [**1**] problem solving [**2**], induction [**3**], abductive scientific (re-)discover [**4**], language translation [**5**] and other cognitive processes ([**1**] [**6**]). This paper describes several evaluations of the Dr Inventor [**7**], which (we believe) is the first analogy-based model to function directly on scientific publications.

The Dr Inventor [**7**] system was developed with the specific objective of identifying creative [**8**] analogies between publications from the discipline of computer graphics. The primary focus of Dr Inventor is to identify similarities between graphics publications such that, when these are presented to computer graphics experts will (frequently) cause creative insight in the user, by highlighting some un-noticed similarities. Dr Inventor is focused on identifying analogies between a user's publication and other papers that typically arise from a different topic (and year) within computer graphics. Early results show that the similarities identified by Dr Inventor will almost always suggest novel and identified source papers that generally would not be read by the user.

As well as being a tool to inspire its users' creativity, Dr Inventor aims to assess the novelty of a submitted document in relation to the other documents contained within its corpus. For example, its users may wish to assess the novelty of an Abstract before writing the full paper. Alternatively, a novice author may write the Abstract of a paper and then use Dr Inventor to identify a similar publication from a different topic, using this paper as a guide to writing their own full paper.

This paper assesses Dr Inventor on challenges related to identifying highly similar or quite similar documents. For example, we wish to assess its ability to quantify the similarity between different versions of the same document. Our focus in this paper is on the metrics used by Dr Inventor and how well they quantify the similarity between highly similar documents and even different versions of the same document. So this paper represents an evaluation of the system at a task that differs from its primary objective. However, the first result we shall discuss relies on human expertise of senior researchers to perform the evaluation.

The paper begins with a brief overview of approaches to retrieving similar texts. We then describe a model of analogy-based similarity before describing the Dr Inventor model for discovering novel and useful analogies between computer graphics publications. Our evaluation and results are then presented in three parts: 1) expert evaluation of the two creative analogies discovered from a corpus of papers from the SIGGRAPH[2] conference series. 2) evaluation of machine generated forward-backward translations 3) evaluation of results for a plagiarism corpus. The paper finishes with some general remarks and conclusion on the evaluation of Dr Inventor.

Document Comparison

Identifying similarities between text-based documents has long been the subject of interest to artificial intelligence. Many approaches have been explored, with some of the more popular approaches being TF-IDF [**9**], LSA [**10**] and many others with many of these approaches being based on word distribution based document representations. Some of the inherent problems with such approaches are discussed in [**11**].

An alternative approach to graph-based document similarity is described in [**11**]. Our approach differs from this in a number of specific regards. Firstly, Dr Inventor's graphs are derived from the output produced by this GATE parser, whereas [**11**] does not use a parser. We do not use external resources to expand the information contained within a document, using the documents as they are presented to perform the similarity assessment. Dr Inventor is based on a cognitive model of figurative thinking, aimed at identifying similarities that are arguably even more abstract and those identified by [**11**]. Our approach looks for figurative similarities that are variously referred to as metaphors, analogies or conceptual blends.

Analogy and Conceptual Blending

[1] Computer Science department, Maynooth University, Ireland, email: donny.hurley@nuim.ie

[2] http://www.siggraph.org/

The approach explored and evaluated in this paper is derived from a cognitive model of people's ability to think figuratively, using two distinct systems of information. At its heart lies the computational model of Gentner's [**12**] influential Structure Mapping Theory (SMT), which posits that many figurative comparisons are best understood by identifying the largest common sub-graph between two systems of information. SMT is a 2-space model that explains why two semantically different concepts can be placed in correspondence between two documents, in SMT it is the topology of information that becomes the prime driver in determining the degree of similarity between two documents - this point shall be highlighted later.

Consider the top and bottom rows of the following image as two distinct abstract diagrams. The problem is to identify the equivalent of the indicated circle from part a) within part b) of the image. If we focus on the circle in isolation and identify the most similar object in part b) then we would identify the central circle from part b). However, if we focus on the relations between object and think of the circle as the right-most object in a sequence, then we would identify the equivalent of the circle from part a as a square in part b.

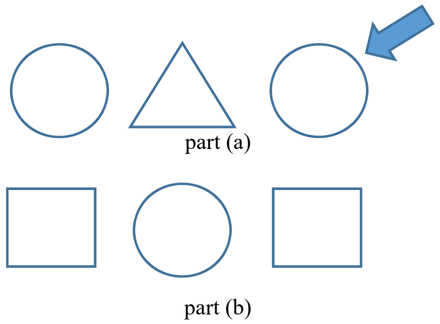


Figure 1: Which object in part (b) is analogous to the indicated circle from part (a)?

This is just a simple illustrative example of the type of reasoning that underlies analogical thinking and conceptual bending.

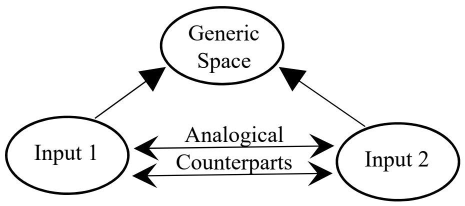


Figure 2. A 3-Space model of Blending

Dr Inventor incorporates SMT into a partial implementation of conceptual blending (or Conceptual Integration Networks) [**13**,**14**]. We use conceptual blending theory to extend our 2-space model of analogy, introducing the generic space that represents the abstract commonality between two mapped (and potentially semantically different) concepts or relations. The implementation of SMT used to identify the counterpart projection between the inputs, basing the counterparts on the analogical similarity between them (forming what we call analogical counterparts)..

A figurative comparison that is common in some cultures compares using your legs (for walking) to a bus (specifically, the "number 11 bus"). This comparison might cause "leg" to be mapped to the concept "bus". The lexical database WordNet [**15**] might identify an abstract connecting concept, identifying both instances of "instrumentality". This abstract concept may then be stored in the generic space and may additionally contribute to evaluating the degree of similarity between the mapped concepts.

Figure 2 outlines the structure of the information used by the central graph-based comparison process of the Dr Inventor model. Dr Inventor compares two text based documents not in terms of their raw textual contents, but instead uses a structured representation derived from a dependency parse of that lexical data – using a parser that has been specifically tailored to the needs of this project.

Firstly, Input 1 and Input 2 are rich, highly structured and complex representations of the contents of the two input documents. Their generation will be outlined in section 4 of this paper. Secondly, the analogical counterparts are identified using an implementation of Structure Mapping Theory [**12**] as supported by the VF2 model of graph matching [**16**]. The output of this mapping phase is a list of paired items between Inputs 1 and 2, based primarily on the *structure* of their representations. Finally, the set of paired items (which may not necessarily be the most semantically similar items) are evaluated by identifying the Generic Space that connect each pair of items, using the WordNet lexical database and the *Lin* [**17**] metric. So the evaluations presented in this paper do not look at the Blended Space, but merely assess the level of similarity that already exists between Inputs one and two.

Dr Inventor

In this section we describe how information is processed through the Dr Inventor [7] system and the results that are found.

Input Data

The Dr Inventor system has as its input a Research Object (RO) [**18**] which, for our purposes, are text based documents. The system is focused on the domain of computer graphics and primarily processes academic papers in this domain. However, an RO can be different types of documents such as psychology material, patents or any other form of text based information. In this paper we will describe the processing that occurs with an academic document and this processing can be performed on any text based documents.

Generating the Research Object Skelton (ROS) Graphs

The Dr Inventor Analogy Blended Creativity (DRI-ABC) model does not work on the RO directly, so for the analogy part of the overall Dr Inventor system we first must process a document and create a Research Object Skelton (ROS). A ROS is an attributed relational graph that contains the core information from a document. At the core of a ROS is the Noun-Verb-Noun type of relations (or Concept-Relation-Concept) and this enables the application of Structure Mapping Theory [**12**] of analogy formation. This requires the extraction of the text based information and so the first step required in processing the document is Text Mining.

Text Mining Framework

A RO is typically in the form of a paper in PDF or text format. To generate a ROS it is necessary to extract the different words, find the dependency relations between the words and attach part of speech tags to each word. Additionally, PDF documents introduce further problems in simply extracting sentences; problems arising from the layout, text flow, images, and equations contained within the PDF.

The extraction of subject-verb-object triples from the textual contents of papers is supported by the Dr Inventor Framework [**19**]. This pipeline of scientific text mining modules is distributed as a stand-alone Java library[3] that exposes an API useful to trigger the analysis of articles as well as to easily retrieve the results. For PDF papers the pipeline invokes the PDFX online Web service[4] [**20**] where the paper is converted into an XML document. Core elements such as the title, authors, abstract and the bibliographic entries are identified.

The Noun-Verb-Noun structure is found within individual sentences and sentences are identified by a Sentence Splitter specifically customised to the idiosyncrasies of scientific discourse. For each sentence, a dependency tree is built using a customised version of [**21**], a Citation-aware dependency parser. The dependency tree identifies types of words (Noun, Verb, Adjective etc.) as well as the types of relationships (subject, object, modifier of nominal etc.). These are used to build the Noun-Verb-Noun structure for the ROS. Additionally the framework identifies co-referent chains in the document, identifying co-referencing words possibly across sentences. This address issues with words such as *it, he, she etc.*

Another feature of the framework is a trainable logistic regression Rhetorical classifier was developed which assigns to each sentence of a paper a rhetorical category (i.e. Background, Approach, Challenge, Outcome and Future Work) used in gold standard manually annotated Dr Inventor Corpus [**22**]. Rather than attempting to find analogies between full papers, the rhetorical categories may be used to find analogies in smaller sections of papers, for example is there an analogy between the *background* of one paper and the *background* of another.

ROS Generation from Text Mining Framework Results

The ROS is constructed by considering the dependency tree formed for each sentence in the publication. As in Agarwal et al. [**23**] a set of rules is applied to these trees, generating connected triples of nouns and verbs. One of the key properties of the ROS graphs is that multiple mentions of the same concept are uniquely represented. This is done either from the co-reference resolution of the text mining framework or by simply joining nodes that have the same word. Relation nodes, i.e. the verbs, can appear multiple times in the ROS.

Each node has an *attribute* of “type” (i.e. noun, verb) and nodes are “tagged” with the rhetorical categories as discussed in the previous section. The format of the ROS was chosen to allow relationships between relations, i.e. second-order or causal relationships between nodes. In the future, when causal relationships are identified by the Text Mining Framework, these nodes will be included in the ROS. The graph database Neo4j[5] uses attributed relational graphs as its representation and as such Dr Inventor uses it for storage of the ROS.

Finding Analogous Document

After storing a collection of ROs in the form of ROS graphs, we want to find the most analogous paper given a chosen target paper. We achieve this by finding (and rating) mappings for the target paper with every other paper contained in the database and choosing the mapping with the highest score. We will now discuss briefly how a mapping is found between one pair of papers.

ROS Mapping

The generated mapping adheres to Gentner’s structure mapping theory [**12**] and its systematicity principle and 1-to-1 mapping constraint. Mapping rules and constraints discussed in [**24**] incorporating both structural mapping and semantic aspects are also utilised. We say that a source graph and a target graph are mapped.

Firstly, the structural mapping between two ROS graphs is based on: 1) graph structure, 2) conceptual structure. Graph structure focuses on identifying isomorphic graphs. Specifically, find the largest isomorphic subgraph of the target in the source. Conceptual structure addresses the conceptual similarity between the nodes and edges that are to be paired by the mapping process [**2**,**25**]. A customised version of the graph matching algorithm VF2 [**16**] is used along with three chosen constraints on the mapping.

Secondly, semantic similarity is used during the computation and the selection phase of candidate pairs. Whenever we encounter two or more candidate pairs that satisfy the structural constraints, we select the pair with the greatest semantic similarity. This similarity is calculated by dictionary-based approach, utilizing the Lin similarity measure [**17**], which in turn uses WordNet [**15**] to calculate the similarity between a pair of source and target nodes of similar type (s, t).

The combination of structural constraints and the preference for mapping semantically similar nodes (where possible) leads to a surprisingly swift mapping process. We conducted a test involving several hundred graphs each involving several hundred nodes, with each being mapped to a clone of itself. Optimal mappings were generated on 100% of these clone-mapping problems, with an average time of under 1 second each on a standard desktop computer. The efficiency of this mapping process plays a significant part in enabling our search for analogically similar document-graphs.

Mapping Metrics

To select the most analogous source paper for a given target paper we must have some way to rate the mappings. We use a unified metric that combines a structural similarity score with a semantic similarity score to have an overall Unified Analogy Similarity (AS).

Jaccard’s coefficient [**26**] is used to measure the structural similarity. The coefficient is used to measure the similarity between two finite sets. The mapping between two graphs is effectively the intersection between the two sets of nodes for the source and the target. As such, the Jaccard’s coefficient can be applied. The

[3] The Dr Inventor Text Mining Framework Java library can be downloaded at: http://backingdata.org/dri/library/

[4] http://pdfx.cs.man.ac.uk/

[5] http://www.neo4j.com

Jaccard's coefficient has a value between 0 and 1, where if it has value 1 the two ROS graphs are identical and if it is 0 then there is no mapping between the two ROS graphs. Jaccard's coefficient gives an estimate of *how much* of the graphs have been mapped.

For the semantic similarity score we use the same Lin metric as used in the semantic mapping. The *Lin* metric always gives a value between 0 and 1. We calculate the overall semantic similarity of the mapping by getting the average semantic similarity of all paired items in the mapping.

The Unified Analogy Similarity score is calculated by multiplying the Jaccard's coefficient by the Semantic Similarity score giving a value between 0 and 1. After finding the scores for mappings of all source papers with a given target paper, we select the most analogous source paper by whichever has the highest unified analogy similarity.

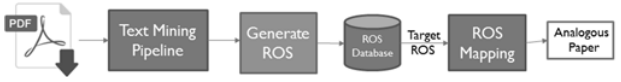


Figure 2. Dr Inventor Paper Processing System

Additional Processing

The above has described the analogy component of the Dr Inventor system. Further processing is done on Computer Graphics papers as part of the overall system. Information is extracted such as topic lists, key words, links between citations, visualisation of similarity between documents and more, as well as a user interface is done by the system, however, this is outside the scope of this paper which is focused on the analogy part of the process.

Finding and Evaluating a Computer Graphics Analogy

To test and evaluate the DRI-ABC system we created a corpus of 957 papers from the SIGGRAPH computer graphics conference. This is one of the top ranked computer graphics conferences in the world. These papers went through the Text-Mining Framework and the ROS generation components of the Dr Inventor system. Papers were included from the years 2002 to 2011 and included many different sub-topics within this discipline. A small and random selection of papers were chosen to serve as target papers and we found analogous source papers. In this section, we will discuss two of the analogies found. The mapping was performed between the (lexical) *abstract* of each paper combined with the rhetorical category of *background* for each paper and performing the mapping only between these sections of each paper.

In generating the two analogies discussed below, all other papers in the corpus were mapped with the target. From the resulting 956 analogies, the analogy metrics were used to choose only the best source analog for the presented target. Early testing showed that there is frequently an exponential distribution in the quality of the analogical comparisons we discovered (as quantified by the analogy metrics). For this and other reasons, we do not expect Dr Inventor to always find creative analogies for a presented paper. So, in this paper we discuss the two best analogies discovered from a list of the 10 best analogies discovered by Dr Inventor.

We will first briefly discuss the Target Paper and what the paper is about. Then we will briefly discuss the Source Paper that was chosen by the system. Finally we will talk about feedback from the analogy. This will be qualitative feedback from a senior professor in computer graphics and then quantitative ratings from multiple computer graphics researchers. Each evaluator spent around 50 minutes evaluating each analogy and they were rated on three properties, on a scale from 1-5, 1) novelty 2) usefulness and 3) challenging the normal view of the topic.

Agreement between raters was calculated using Krippendorff's

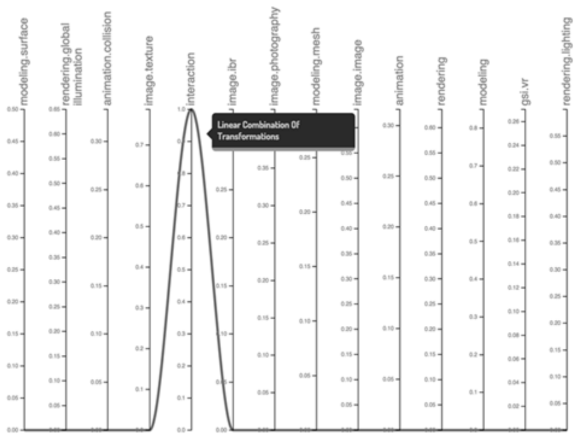


Figure 3. Target Paper 1 Topics

alpha, as the rating scale formed a numeric interval (1-5) with small differences (4 -5) being of less significance than larger differences (1-5) on this linear numeric scale. Analysis of the rating data for 12 rates using a 5 point Likert scale returned the following Krippendorf's alpha values:

(1) Novelty of 0.344,
(2) Usefulness 0.274
(3) Challenge the norms 0.394

The might be considered a surprisingly high level of agreement, given that creativity is often said to be very subjective – particularly given the diversity in the experience possessed by the different raters.

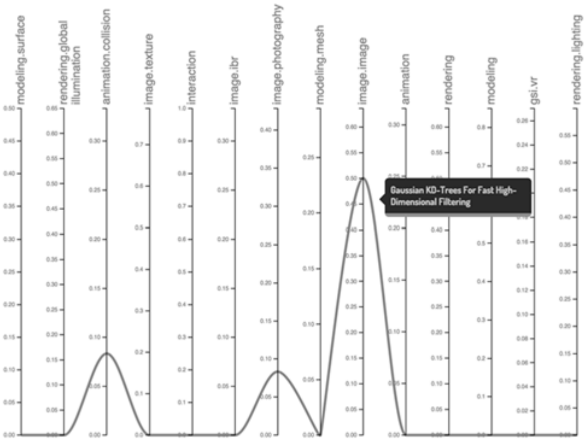


Figure 4. Source Paper 1 Topics

First Analogy

Target Paper

The target paper we will discuss is "*Linear Combination of Transformations"* by Marc Alexa which appeared in SIGGRAPH 2002. A brief description of the paper is: This paper's problem is trying to transform a 3D model. The problem is that transforming a 3D model is based on matrix or quaternion operations and these operations are not commutative. The proposed solution is to break each transformation matrix into smaller parts and perform them alternatively and thus the linear combination of smaller matrix transformations is closer to being commutative. Figure 2 shows the topics the paper is contained within (Interaction). This image is generated by Dr Inventor.

Source Paper

Searching through the full corpus of 957 papers, the paper chosen with the highest Analogy Similarity score was "*Gaussian KD-Trees for Fast High-Dimensional Filtering*" by Andrew Adams et al which appeared in SIGGRAPH 2009. A brief description of the paper is: The paper presents an algorithm to accelerate a broad class of non-linear filters. The problem is non-linear filters scale poorly with filter size. The proposed solution it to propose a new Gaussian *kd*-tree, which sparsely represents the high-dimensional space as values stored at points. Figure 3 shows that the paper is contained in the topics: *Image.photography, image.image*and *Animation.Collision*.

Analogy Feedback

A senior professor in Computer Graphics examined these two papers after the system identified them. He considered the two papers to be very analogous and promising. As part of the mapping, the term "matrices" in the target paper was mapped to the term "filter" in the source paper. This suggested that the manipulations applied to matrices can be applied to filters and vice-versa. To show how Dr Inventor could be applied as a Creativity Support Tool this suggested new research ideas that could be further explored. Such as, can we break down image filters into small parts and perform them alternately as was done to the matrices in the analogous paper. Or cascade image filtering and their commutativity.

Two of the interesting things about the found analogy are the differences in the year (2002 and 2009) and also the topics each paper is contained in. They are somewhat dissimilar. This suggests the papers would not usually be compared to one another and they would not typically be papers read when trying to find analogous problems. Dr Inventor is identifying structures not normally considering when trying to find similar papers. Furthermore the conceptual similarity (the semantic similarity between mapped nouns) is 0.37 showing a marked difference between the concepts while a high relational similarity (0.79) was found.

Additionally evaluation of the analogy was performed by 13 evaluators, mostly post-graduate students in computer graphics but also post-doctoral researchers and two senior professors. The average ratings obtained were 4.5 for novelty, 3.7 for usefulness and 4.1 for challenging the normal view of the topic.

Second Analogy

Target Paper

The second target paper is "*Fast Bilateral Filtering for the Display of High-Dynamic-Range Images*" by F Durand and J Dorsey from SIGGRAPH 2002. This paper presents a technique for the display of high-dynamic-range images, which reduces the contrast while preserving details and how poor management of light – under- or over-exposed areas, light behind the main character, etc. – is the single most-commonly-cited reason for rejecting photographs. It has the topics Image Processing and Photograph.

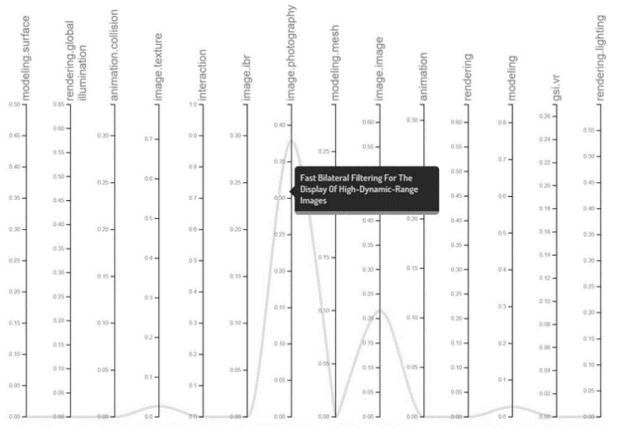


Figure 5. Target Paper 2 Topics

Source Paper

The paper with the highest Analogy Similarity score was "*Curve Skeleton Extraction from Incomplete Point Cloud"* by A Tagliasacchi, H Zhang and D Cohen-Or from SIGGRAPH 2009. This paper presents an algorithm for curve skeleton extraction from imperfect point clouds where large portions of the data may be missing. The problem arises from incomplete data during 3D laser scan. The point cloud data contains large holes. The paper has the topics Modeling and Point Cloud.

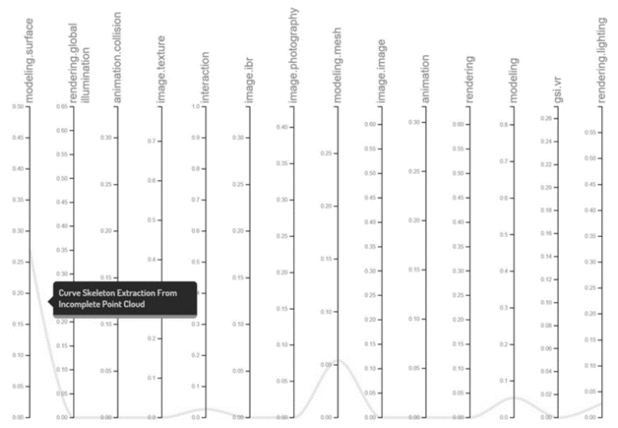


Figure 6. Source Paper 2 Topics

Analogy Feedback

A different senior professor provided the qualitative feedback for this analogy. Each paper, when broken down to its basics, is discussing about "missing data" in the image. In the case of the target paper, data about the image is obscured by the contrast of a digital photograph as it cannot as accurately capture the image as the human eye. In the source paper, data points of the 3D image are blocked from being scanned by the lasers. Mappings are found between the term "Hole" in the target paper with "Area" in the source paper. That is "the photo will contain under- and over-exposed areas" is mapped to "data contain large holes caused during 3D laser scan", so Dr Inventor can suggest the similarities between the two paper problems.

The results of this analogy suggested to the professor several possible new ideas for reconstruction of hidden information. How would similar techniques apply to motion capture, missing video data and more.

As in the first example the two papers are found many years apart and the topics they are contained within are not similar. Again, Dr Inventor is finding far analogies that typically would not be found by a normal literature review when attempting to write a research paper. The conceptual similarity was again low (0.37) while the relational similarity was high (0.8).

For the evaluation performed by more researchers, the average ratings were obtained for the same three categories. 4.1 for novelty, 3 for usefulness and 3.3 for challenging norms.

Further Usage of System

We have described the usage of PDF academic papers through the Dr Inventor system and two of the results found. Additionally, Dr Inventor can be expanded outside its original focus on the domain of computer graphics. ROS graphs can be formed from any text based documents and commonly used plain text files can be processed through the system. We now discuss some of these specific formats that can be used.

We describe the evaluation of Dr Inventor on two tasks that lie beyond the initial scope of this project. Firstly we assess Dr Inventor and particularly its similarity metrics at the task of automatically evaluating the faithfulness of machine translation services. Secondly, we assess it at the task of detecting the degree of similarity between a document and plagiarised versions of those documents. In this section we focus our evaluation on aggregations of results rather than presenting individual comparisons.

Machine Translation Evaluation

Another means of evaluating the DRI-ABC system is to evaluate translations generated by machine translation services. So, this section represents a joint evaluation of DRI-ABC as well as the machine translations themselves. This is searching for a near analogy i.e. generating similar but slightly different versions of the document.

By taking an original document (in English), translating it to the chosen language and then translating this back (to English) we can check for similarities between the original document and the translated back document. One advantage of our graph matching approach is that it is not sensitive to the introduction (or removal) of sentence boundaries between the original and back-translated documents.

Corpus of Translated Documents

The psychology dataset was collected from psychology literature [**2**] on analogical reasoning and problem solving, consisting of 36 English texts used in several human-subject tests. These texts represent stories containing between 50 and 400 words (average=205) with several being in the form of analogous pairs of stories. A selection of documents (18) from this dataset was translated into different languages and then back-translated to English. Google Translate was used to perform the translations and this translation corpus was created specifically to contribute to the evaluation of Dr Inventor. By varying the difference between English and the target language we aim to evaluate the metrics used by Dr Inventor. Our expectation before undertaking this work was that, as the target language became more distant from English the similarity score between the original and back-translated text should decrease.

The languages chosen were *Irish, Russian, Spanish, French, German, Arabic* and *Amharic*. These languages were selected due to feedback received from native speakers of these languages on Google Translate and as Dr Inventor project members are (mostly native) speakers of these languages. It is expected that Spanish, French and German will be ranked the highest, while Arabic and Amharic will be ranked the lowest. Native speakers of Arabic and Amharic read some sample documents (not part of any Dr Inventor corpus) and they were generally rated as being of poor quality by these speakers. In particular, Amharic was only added to Google Translate in early 2016 and as such, it has not had as long a time to train and refine the translation system. Additionally the languages Russian and Irish were also selected to see if they could be evaluated. Spanish and French are Romance languages with well-developed machine translation systems, so our expectation was that these would produce some of the most faithful translations.

Of course our evaluation will also qualitatively discuss the maturity of Google's translation service for each language.

Translation Quality Estimation

The best results on the corpus were produced, as expected, for the languages Spanish, French and German. As these are the languages most closely related to English and they are also some of the most widely used and well-developed translation systems. It was decided to use these scores and results to be a baseline for a good translation score. Native English speakers compared the original document with the back-translated document they were generally considered to be fairly accurate re-representations of the original text.

Running the system using the Arabic and Amharic languages also produced the expected results as the scores received were much lower than the "well translated" languages. Native English speakers comparing the original document with the back-translated document agreed that numerous errors did occur. As discussed above, native speakers of these languages did find errors and problems. These were not unexpected due to the dissimilarity in the languages themselves. In particular, in Arabic the word order can be quite different even due to the differences in direction of reading. Additionally, in Arabic, the subject could be dropped from the sentence but still have the same meaning, as the subject is implicitly understood.

Finally, the system was run with our two "testing" languages, Irish and Russian. By using the baseline of the "well translated" languages and the "badly translated" languages, it showed that the

Google translate system worked quite well with the Russian and Irish. Their scores were not as high as Spanish, French or German but they were much better performing than Arabic and Amharic.

The box plot below (Figure 7) summarises all the results of this translation evaluation. Overall it showed the Dr Inventor system performed as expected at evaluating the "well-translated" and "badly-translated" languages.

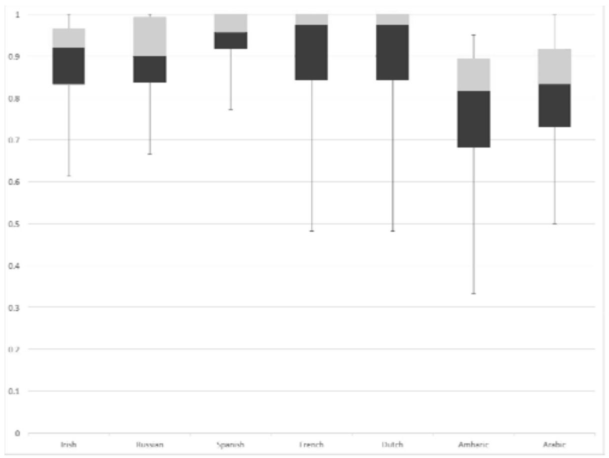


Figure 7. Similarity scores for the languages Irish, Russian, Spanish, French, German, Arabic and Amharic

Plagiarism Corpus

A corpus of plagiarised short documents was created [**27**] with the aim that it could be used for the development and evaluation of plagiarism detection tools. The corpus consists of short answers to computer science questions and the plagiarism challenge has been simulated, representing various degrees of plagiarism. Using this corpus we assessed Dr Inventor's ability to detect plagiarism among these documents, i.e. searching for near analogies.

Levels of Plagiarism

Each answer used a Wikipedia entry as a source text. The corpus has four levels of plagiarism:

1) *near copy*: simply copying text from the entry.
2) *light revision*: basing the answer on the entry but the text could be altered in basic ways. Words could be substituted and paraphrasing could be used.
3) *heavy revision*: again basing the answer on the entry but the text was rephrased using different words and structure.
4) *non-plagiarism*: by using standard learning materials answers were constructed by using the participants own knowledge.

Corpus Contents

19 participants were asked to answer 5 questions according to the guidelines of the level of plagiarism to be used. 95 answers were generated by these students. Including the original Wikipedia entry 100 documents are contained within the corpus and these documents are passed through the Dr Inventor system to see how it assesses the four different levels of similar contained within this corpus.

Output from the System

All of the 100 documents were processed by the Dr Inventor system and ROS graphs were created for each of them. The original document was compared against the 4 different plagiarised versions by mapping their respective ROS graphs. The semantic similarity score from Dr Inventor was measured and the following box plot was obtained over the corpus.

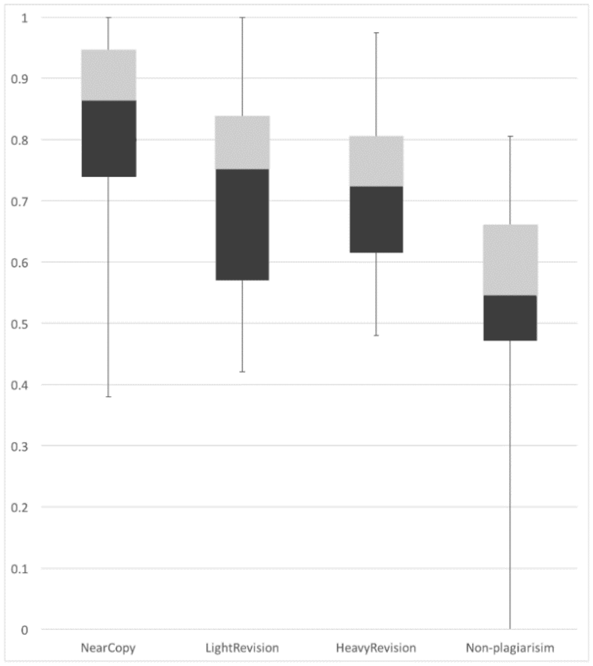


Figure 8. Semantic Similarity for Different levels of Plagiarism

This shows that as the amount of plagiarism decreases, the semantic similarity found by Dr Inventor decreases as well. This again was a very pleasing result as it shows that the metrics currently in use by Dr Inventor show a degree of refinement in estimating the similarity between plagiarised versions of documents.

Future Work

Our earlier results show that the existing metrics used by the Dr Inventor system appear to operate effectively, even when there's relatively little semantic distance between the two input documents. This gives us confidence to start exploring its use in dealing with patent applications. Estimating the similarity between patent applications [**28**] is particularly important to Dr Inventor. One current undertaking relates to adapting the parser to correctly handle some of the lexical peculiarities of patents so that they are correctly processed by the parser [**29**].

Some future work is based on the notion that many commercially sensitive patents are written such that they will not be found by existing retrieval tools. This makes the challenge of filing a defence against a new patent application very difficult for the holder of an existing patent. In future work we hope to be able to identify some of these patents.

Conclusion

We described the Dr Inventor system that identifies figurative and structure-based similarities between text based documents.

The first evaluation used Dr Inventor's metrics to identify two very high quality analogies between publications from the SIGGRAPH conference series. Each was evaluated by a senior researcher in computer graphics and this showed that each comparison was both novel and also represented a reasonable hypothesis that was worthy of their consideration. Both evaluators agreed that each could (at least) be considered as the basis for subsequent research.

The second evaluation of Dr Inventor outlined a translation corpus that was based on English language versions of 18 different texts sourced from various psychological studies on the analogy process. These texts were translated into seven different target languages and then back-translated to English, creating different versions of the source document. Dr Inventor showed that the closest languages produced the best translations as estimated by its metrics. Similarly the newest translation languages which are also the most distant from English produced the lowest results - the two intermediate languages are producing intermediate results. While lacking a certain degree of refinement these results show that Dr Inventor may be usefully used to estimate the quality of "roundtrip" translations.

Dr Inventor and its similarity metrics were also assessed at the task of evaluating a "short answers" plagiarism corpus. This contained short documents in one of three levels of plagiarism, as well as one non-plagiarised version of each document. Again this evaluation showed that Dr Inventor showed a good ability to identify between the different levels of plagiarism within the corpus.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme ([FP7/2007-2013]) under grant agreement no 611383.

REFERENCES

[1] K., J. Holyoak, D. Gentner, B.,N. Kokinov, D. Gentner, and K.,J. Holyoak, "Introduction: The place of analogy in cognition," in *The Analogical Mind: Perspectives from cognitive science*., 2001, pp. 1-19.

[2] M.,L. Gick and K.,J. Holyoak, "Analogical problem solving," *Cognitive psychology*, vol. 12, no. 3, pp. 306-355, 1980.

[3] M. L. Gick and K.J. Holyoak, "Schema induction and analogical transfer," *Cognitive psychology*, vol. 15, no. 1, pp. 1-38, 1983.

[4] D.P. O'Donoghue and M.T. Keane, "A Creative Analogy Machine: Results and Challenges," in *4th International Conference on Computational Creativity*, Dublin, Ireland, 2012, pp. 17-24.

[5] P. Langlais and Yvon F., "Scaling up Analogical Learning.," in *Proceedings of the International Conference on Computational Linguistic*, 2008, pp. 49--52.

[6] Prade Henri and Richard Gilles, *Computational Approaches to Analogical Reasoning: Current Trends*., 2014, vol. 548.

[7] D.P. O'Donoghue, Y. Abgaz, D. Hurley, F. Ronzano, and H. Saggion, "Stimulating and Simulating Creativity with Dr Inventor," in *6th International Conf on Computational Creativity*, Utah, USA, 2015.

[8] M. A. Boden, "Creativity and artificial intelligence," *Artificial Intelligence*, vol. 103, no. 1, pp. 347-356, 1998.

[9] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513-523, 1988.

[10] T. K. Landauer, P. W. Foltz, and D. Laham, "An Introduction to Latent Semantic Analysis," *Discourse Processes*, vol. 25, pp. 259-284, 1998.

[11] C. Paul, A. Rettinger, A. Mogadala, C. A. Knoblock, and P. Szekely, "Efficient Graph-Based Document Similarity," in *International Semantic Web Conference*, May, 2016, pp. 334-349.

[12] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive Science, Volume 7*, pp. 155-170, 1983.

[13] Gilles Fauconnier and Mark Turner, "Conceptual integration networks," *Cognitive science*, vol. 22, no. 2, pp. 133--187, 1998.

[14] T. Veale, D. P. O'Donoghue, and M. T. Keane, "Computation and Blending," *Cognitive Linguistics 11 (3/4)*, pp. 253-281, 2000.

[15] G. A. Miller, "WordNet: A Lexical Database for English.," *Communications of the ACM*, vol. 38, no. 11, pp. 39-41, 1995.

[16] L.,P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," in *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, 2004, pp. 1367-1372.

[17] Dekang Lin, "An Information-Theoretic Definition of Similarity," , San Francisco, CA, USA, 1998.

[18] K. Belhajjame et al., "Workflow-centric research objects: First class citizens in scholarly discourse," in *9th Extended Semantic Web Conference*, Hersonissos, Greece, 2012.

[19] F. Ronzano and H. Saggion, "Knowledge Extraction and Modeling from Scientific Publications," in *In the Proceedings of the Workshop "Semantics, Analytics, Visualisation: Enhancing Scholarly Data" co-located with the 25th International World Wide Web Conference*, Montreal, Canada, 2016.

[20] A. Constantin, S. Pettifer, and A. Voronkov, "PDFX: fully-automated PDF-to-XML conversion of scientific literature," in *Proceedings of the 2013 ACM symposium on Document engineering*, 2013, pp. 177–180.

[21] B. Bohnet, "Very high accuracy and fast dependency parsing is not a contradiction.," in *Proc. 23rd International Conference on Computational Linguistics*, 2010, pp. 89–97.

[22] Beatriz Fisas, Francesco Ronzano, and Horacio Saggion, "On the Discoursive Structure of Computer Graphics Research Papers," in *The 9th Linguistic Annotation Workshop held in conjuncion with NAACL 2015.*, 2015.

[23] B. Agarwal, S. Poria, N. Mittal, A. Gelbukh, and A. Hussain, "Concept-level sentiment analysis with dependency-based semantic parsing: A novel approach," *Cognitive Computation*, pp. 1-13, 2015.

[24] Keith, J. Holyoak and Paul Thagard, "Analogical mapping by constraint satisfaction," *Cognitive Science*, vol. 13, pp. 295-355, 1989.

[25] Dedre Gentner and Arthur, B. Markman, "Defining structural similarity," *Journal of Cognitive Science*, vol. 6, pp. 1-20, 2005.

[26] Paul. Jaccard, "Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines," *Bulletin de la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 241-272, 1901.

[27] P. Clough and M. Stevenson, "Developing A Corpus of Plagiarised Short Answers," *Language Resources and Evaluation: Special Issue on Plagiarism and Authorship Analysis, In Press.*

[28] S. Reardon, "Text-mining offers clues to success," *Nature*, vol. 509, no. 7501, pp. 410-410, 2014.

[29] A. Burga, J. Codina, G. Ferraro, H. Saggion, and L. Wanner, "The challenge of syntactic dependency parsing adaptation for the patent domain.," in *ESSLLI-13 Workshop on Extrinsic Parse Improvement.*, 2013, August.

:

Playing Atari Games with Deep Reinforcement Learning and Human Checkpoint Replay

Ionel-Alexandru Hosu[1] and **Traian Rebedea**[2]

Abstract. This paper introduces a novel method for learning how to play the most difficult Atari 2600 games from the Arcade Learning Environment using deep reinforcement learning. The proposed method, called *human checkpoint replay*, consists in using checkpoints sampled from human gameplay as starting points for the learning process. This is meant to compensate for the difficulties of current exploration strategies, such as ε-greedy, to find successful control policies in games with sparse rewards. Like other deep reinforcement learning architectures, our model uses a convolutional neural network that receives only raw pixel inputs to estimate the state value function. We tested our method on Montezuma's Revenge and Private Eye, two of the most challenging games from the Atari platform. The results we obtained show a substantial improvement compared to previous learning approaches, as well as over a random player. We also propose a method for training deep reinforcement learning agents using human gameplay experience, which we call human experience replay.

1 INTRODUCTION

General game playing is an extremely complex challenge, since building a model that is able to learn to play any game is a task that is closely related to achieving artificial general intelligence (AGI). Video games are an appropriate test bench for general purpose agents, since the wide variety of games allows solutions to use and hone many different skills like control, strategy, long term planning and so on. Designed to provide enough of a challenge to human players, Atari games in particular are a good testbed for incipient general intelligence. With the release of the Arcade Learning Environment (ALE) [3] in 2012, general game playing started to gain more popularity. ALE is an emulator for the Atari 2600 gaming platform and it currently supports more than 50 Atari games. They are somewhat simple, but they provide high-dimensional sensory input through RGB images (game screen).

Deep learning models have achieved state-of-the-art results in domains such as vision [13, 22, 11] and speech recognition [2]. This is due to the ability of models such as convolutional neural networks to learn high-level features from large datasets. Along with these achievements, reinforcement learning also gained a lot of ground recently. These powerful models helped reinforcement learning where it struggled the most, by providing a more flexible state representation. As a consequence, tasks such as learning multiple Atari games using a single, unmodified architecture became achievable through deep reinforcement learning [15, 16]. However, in environments characterized by a sparse or delayed reward, reinforcement learning alone is still struggling. This is caused mostly by naive exploration strategies, such as ε-greedy [21], that fail to find successful policies to discover an incipient set of rewards. This is the case for the most difficult video games from the Atari platform, such as Montezuma's Revenge and Private Eye, that prove to be too challenging for all current approaches.

[1] University Politehnica of Bucharest, Romania, email: ionel.hosu@stud.acs.upb.ro
[2] University Politehnica of Bucharest, Romania, email: traian.rebedea@cs.pub.ro

In a different context, AlphaGo [20], a system combining reinforcement learning with Monte Carlo Tree Search, defeated Lee Sedol, one of the top Go players in the world. Nearly 20 years after Garry Kasparov was defeated by Deep Blue [7], this represented an important milestone in the quest for achieving artificial general intelligence. Together with the launch of the OpenAI Gym [6], it is one of the most significant advances that reinforcement learning made in the last couple of years.

This paper demonstrates that it is possible to successfully use a learning approach on the most complex Atari video games, by introducing a method called *human checkpoint replay*. Our method consists of using checkpoints sampled from the gameplay of a human player as starting points for the training process of a convolutional neural network trained with deep reinforcement learning [15, 16].

The paper proceeds as follows. In section 2, we present the most relevant results on learning for Atari video games. Section 3 follows with a more in-depth analysis and explanation of the current results for Atari games. We also motivate our choice of games for evaluating our architecture, providing a brief description of the difficulties encountered in these games. In section 4 we provide a thorough description of the proposed methods and deep reinforcement learning architecture. Section 5 presents the results of our approach, as well as a detailed discussion on the performed experiments.

2 RELATED WORK

After the release of the Arcade Learning Environment, there have been numerous approaches to general game playing for Atari games. Approaches such as SARSA and contingency awareness [4] delivered promising results, but were far from human-level performance. The use of neuro-evolution [9, 10] on the Atari platform dramatically improved these results, but playing Atari games as well as a human player still seemed unachievable.

The first method to achieve human-level performance in an Atari game is deep reinforcement learning [15, 16]. It mainly consists of a convolutional neural network trained using Q-learning [25] with experience replay [14]. The neural network receives four consecutive game screens, and outputs Q-values for each possible action in the game. Experience replay is used in order to break the correlations between consecutive updates, as Q-learning would prove unstable in an online setting. The most important aspect of this approach is that it can be used to construct agents that do not possess any prior domain knowledge, thus rendering them capable of learning to perform multiple different tasks.

After this first success of deep reinforcement learning [15, 16], a number of improvements have been made to the original architecture. The fact that its convergence was slow and it took multiple days to train a neural network on a single game motivated the development of a distributed version of deep reinforcement learning [17] which reduces the training times and improves the existing results.

Another notable improvement came from the realization that the Q-learning algorithm sometimes performs poorly by overestimating action values [8]. This issue may be solved by employing double Q-learning [23] - using it together with deep reinforcement learning on the Atari domain fixed the overestimation problem that appeared in some of the games.

Another notable approach to Atari game playing is the bootstrapped deep Q-network (DQN) [18], which proposes a novel and computationally efficient method of exploration. Its main contribution is to find an alternative for simple, inefficient exploration strategies, such as ε-greedy. To achieve this, bootstrapped DQN produces distributions over Q-values instead of Q-values. Sampling from these distributions allows the model to renounce using exploration strategies.

The current state-of-the-art on Atari games is achieved using a method called prioritized experience replay [19]. It starts from the assumption that not all the transitions present in the replay memory have the same importance. The agent learns more effectively from some transitions, other ones being redundant, not relevant, etc. The method proposes a prioritization regarding how often transitions are used for updates in the network based on the magnitude of their temporal difference (TD) error [24]. Prioritized replay leads to an improvement in 41 out of the 49 games, delivering human-level performance in 35 of these games.

Although the aforementioned methods brought considerable improvements over the original deep reinforcement learning architecture, there still are some Atari games for which none of the previously published methods are able to learn whatsoever. These games feature a sparse reward space and are more complex in many aspects than the vast majority of games from the Atari platform. We propose a new method called *human checkpoint replay* that, when used together with deep reinforcement learning, is able to learn successful policies for the most difficult games from the Atari platform, thus resulting in significantly improved performance compared to prior work.

3 BACKGROUND

Before proceeding to describe our approach, it is important to perform a more detailed analysis of the games used in our experiments; the discussion focuses on highlighting some of the aspects that make them so challenging. To evaluate our approach, we have chosen Montezuma's Revenge and Private Eye - two of the most difficult games from the Atari platform. Therefore our analysis focuses on these two games; however the main points are also valid for other challenging games where there is no known learning policy which achieves a better score than a random agent.

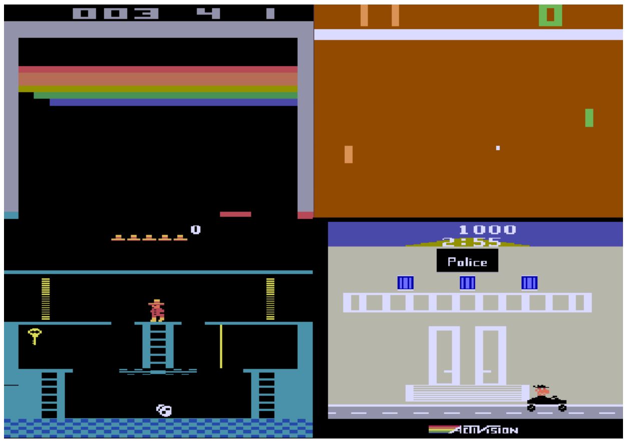


Figure 1. Screen shots from four Atari 2600 Games: (*Left-to-right, top-to-bottom*) Breakout, Pong, Montezuma's Revenge, Private Eye

3.1 Montezuma's Revenge

Montezuma's Revenge is a game from the Atari 2600 gaming console that features a human-like avatar moving across a series of 2D rooms that form labyrinth. In order to advance in the game, the player must move in a consistent manner, jump over obstacles, climb ladders, avoid or kill monsters, and collect keys, jewels and other artifacts which provide a positive reward by increasing the game score. Some of these collectible artifacts grant additional abilities - for example, collecting a key enables the player to open a door upon contact. However, after opening a door, the player loses the key and needs to collect additional keys for opening any other doors. Other collectible items include a torch which lights up dark rooms and swords which kill monsters. The game consists of three levels, each of them containing 24 rooms.

An important characteristic of the game is that collectible items are sparse and thus the reward space is also very sparse. When the game starts, in order to collect the first reward - represented by a key - the avatar is required to descend on two ladders, walk across the screen and over a suspended platform, jump over a monster and climb another ladder. After collecting the key, the avatar needs to return close to the starting point, where two doors are available which can now be opened using the collected key. The player starts the game with five ”lives”, and each time it loses one life, the avatar is respawned in the same room. For optimal play, a memory component is required, as the player does not possess information about other rooms in terms of rewards collected or monsters killed. The only information available on the screen apart from the environment is the game score, the number of remaining lives and the artifacts currently held by the avatar.

All current approaches using deep reinforcement learning fail to learn any successful control policies for Montezuma's Revenge. This happens mostly due to the ε-greedy strategy failing to explore the game in a consistent and efficient manner. Every four frames, a DQN agent has to choose between 18 different actions. Given how, in order to receive the first positive reward, the player is required to perform a complex and consistent sequence of actions, using such a simple exploration strategy makes learning virtually impossible. It is also worth mentioning that the way in which such a simple strategy explores an environment like Montezuma's Revenge does not resemble the way in which a human player does it. This is due to two factors. First, there is a strong correlation between multiple successive actions of a human player in the video game, even when the player does not know how to play the game yet. This is because the exploration exhibited by a human player is not random, but influenced by the current state of the environment. Second, a human player always makes use of commonsense knowledge when dealing with a new learning task. This also influences the manner in which the human player explores a virtual environment, especially when it contains elements resembling real-world objects (e.g. ladders, monsters, keys, doors, etc.) For example, when the game avatar is located on a ladder, a human player will only use the up and down actions without having to learn to do so by exploring the game states. The player already knows, from real life, that other actions are not useful, as they do not lead to other states in the game.

3.2 Private Eye

The second game chosen for our evaluation is Private Eye. It is a game that features an avatar for a private investigator who is driving a car that can move around and jump vertically or over obstacles. The game environment can be seen as a labyrinth as well, as it consists of multiple roads located in a city or in the woods. In the city there are buildings near the roads, some of which play a special role in the game. The objective of the game is to capture thieves that sit behind the windows of different buildings. The thieves appear briefly and the player must move and ”touch” the area near a thief in order to capture him/her. Capturing thieves provides the player with rewards and sometimes with special items, that must be returned to specific buildings (e.g. bags of money need to be returned to the bank, guns returned to the gun store). The player starts the game with 1000 points, but is penalized if he bumps into obstacles like birds and mice, or is attacked by thieves. The concept of multiple lives is not present in this game, however there is a three minute time cap for solving a game level; when the time limit has been reached, the game ends.

The game features similar exploration difficulties present in Montezuma’s Revenge. As a consequence, this game has also proven to be a challenge for current deep reinforcement learning methods. The presence of a memory component for optimal play is even more important, as the player must travel long distances between collecting game items and dropping them at the appropriate locations. Some portions of the game look identical, and there is also a certain order in which the tasks should be carried out. For example, the game features a thief that must be captured and brought to the police building, but this can only be done after all the items in the current level have been returned where they belong. Capturing this final thief provides the highest in-game reward.

3.3 Discussion

The more challenging games from the Atari console (such as Montezuma’s Revenge and Private Eye) present a particular difficulty compared to the simpler ones: the agent is not penalized for standing still. This could be yet another factor that further prevents efficient exploration. In contrast, for games like Breakout or Pong (Figure 1), repeatedly choosing the no-op action will quickly lead to the end of the game or to losing points. The agent can thus learn to avoid this action, as it will be associated with a low utility value.

4 DEEP REINFORCEMENT LEARNING WITH HUMAN CHECKPOINT REPLAY

In this section we provide a description of the methods that we propose for training deep reinforcement learning agents on the Atari domain, as well as the architecture of the convolutional network that we used for the training process.

4.1 Deep Reinforcement Learning

Deep reinforcement learning [15, 16] was the first method able to learn successful control policies directly from high-dimensional visual input on the Atari domain. It consists of a convolutional neural network that extracts features from the game frames and approximates the following action-value function

$$Q^*(a,b) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... \mid s_t = s, a_t = a, \pi] \quad (1)$$

The computed value represents the sum of rewards r_t discounted by γ at each time step t, using a policy π for the observation s and action a. To solve the instability issue that reinforcement learning presents when a neural network is used to approximate the state-value function, experience replay [14] is used, as well as a target network [16]. In order to train the network, Q-learning updates are applied on minibatches of experience, drawn at random from the replay memory. The Q-learning update at iteration i uses the following loss function

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)}[(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i))^2] \quad (2)$$

Here γ is the discount factor determining the agents horizon, θ_i are the parameters of the Q-network at iteration i and θ_i^- are the network parameters used to compute the target at iteration i. Differentiating the loss function with respect to the network weights gives the following gradient:

$$\nabla_{\theta_i} L(\theta_i) = \mathbb{E}_{s,a,r,s'}[(r + \gamma \max_{a'} Q(s',a';\theta_i^-) - \\ - Q(s,a;\theta_i)\nabla_{\theta_i} Q(s,a;\theta_i)]. \quad (3)$$

Our proposed approaches use the deep reinforcement learning architecture from [16]. Almost all the hyperparamaters have the same values, with the exception of the final exploration frame, for which we used a value of $4,000,000$ instead of $1,000,000$. We empirically found that using this value results in a slightly better performance of the agents. This may be caused by the fact that the most difficult games are also characterized by greater complexity and size of the state space, therefore a slower annealing of ε may be helpful. Due to hardware limitations and the amount of time required for training deep Q-networks, we did not test for other values of this hyperparameter. Another structural difference we need to mention is the fact that, for the *human experience replay* method presented next, we employed an additional replay memory.

4.2 Human Checkpoint Replay

For the most difficult games from the Atari platform that are characterized by sparse rewards, the original deep reinforcement learning approach [15, 16] is not able to achieve positive scores. Thus the agents trained using deep reinforcement learning perform no better than a random agent. As seen in the previous section, these games start in a state from which reaching the first reward is a long and challenging process for any player which does not possess any prior knowledge (such as commonsense world knowledge). Also considering the 18 actions available to the agent in each state, the ε-greedy strategy fails to find any game paths to a first state with positive reward. This hinders the convolutional neural network to learn relevant features to separate reward-winning states and actions from the bulk. Drawing inspiration from curriculum learning [5] and the human starts evaluation metric used for testing Atari agents [17], we introduce the *human checkpoint replay* method. This consists of generating a number of checkpoints from human experience in the Arcade Learning Environment [3] and storing them to be used as starting points for the training of deep reinforcement learning agents. Instead of resetting the environment to the beginning of the game each time a game episode ends, a checkpoint is randomly sampled from the checkpoint pool and restored in ALE.

The intuition behind this approach is that at least some of the checkpoints will have a reward located close enough in the game space for the ε-greedy strategy to be able to reach it. This way, the convolutional neural network is able to learn relevant features from the game frames and then successful control policies. As the training process advances, these will help the agent to become gradually more capable to reach rewards that are located farther away from the start state. Our method can also be thought of as being related to the planning concepts of landmarks [12] and probabilistic roadmaps [1].

4.3 Human Experience Replay

As a possible solution to the inefficiency of ε-greedy exploration in sparse reward environments, we also proposed training a deep reinforcement learning agent using offline human experience, combined with online agent experience. We dubbed this approach *human experience replay*. It consists of storing human gameplay experience in same form of (s, a, r, s') tuples in a separate replay memory and using it along with the original replay memory containing agent experience. This is meant to provide the agent with training samples that result in a positive reward, therefore making learning possible in environments that feature a sparse reward signal. The training process consists of repeatedly sampling a minibatch composed of both human transitions and agent transitions.

5 EXPERIMENTS

This section provides a thorough description of the experiments performed using our proposed approach for the two selected Atari games, Montezuma's Revenge and Private Eye. As mentioned earlier, we have chosen these two games because they are among the most challenging games on the Atari platform, even for human players. At this point no computer strategy has been able to learn an exploration technique which is better than a random agent, mainly due to the fact that no strategy is able to learn a solution which is able to reach any reward in the game. Therefore, any progress made towards solving these games will provides useful insights in the quest of developing general purpose agents.

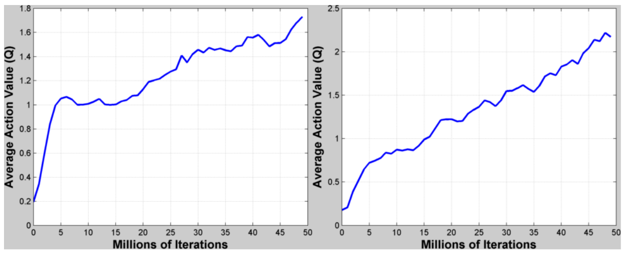


Figure 2. The two plots show the average maximum predicted action-value during training for our HCR DQN method on Montezuma's Revenge *(left)* and Private Eye *(right)*

5.1 Human Checkpoint Replay

The Arcade Learning Environment provides the capability of generating checkpoints during gameplay. These make it possible to continue running an environment from a given state at a later time by restoring a specific checkpoint in the emulator. The checkpoint consists of the memory content of the Atari 2600 console.

For the human checkpoint replay method (HCR DQN), we generated 100 checkpoints from a human player's experience for each game, stored them in an external file and then used them as starting points for the environment at training time, as well as for testing. The checkpoints that we used for training and testing, as well as the code for training deep reinforcement learning agents with human checkpoint replay, are publicly available [3]. We trained our networks using the generated checkpoints and performing Q-learning updates as describe in [16] for 50 million frames on each game. The two plots in Figure 2 show how the average predicted Q evolves during training on the games Montezuma's Revenge and Private Eye.

As discussed in section 3.3, in difficult games such as Montezuma's Revenge and Private Eye, the avatar is not penalized for repeatedly choosing the no-op action. This raises two major issues. First of all, efficient exploration is prevented due to the neutral effect of repeatedly taking the no-op action. Also, in a deep reinforcement learning setting in which experience replay is used, by repeatedly choosing the no-op action, the replay memory will consistently be filled with transitions that are not relevant for the learning process. In order to avoid this outcome, we limit each training episode to 1800 frames, corresponding to 30 seconds of gameplay. By doing this, we make sure that the replay memory is populated with transitions that are relevant for the training process, as the agent will eventually be placed in checkpoints from which rewards are more easily accessible.

5.2 Human Experience Replay

Using ALE, we generated 1.2 million frames of human experience (about 5.5 hours of gameplay) for Montezuma's Revenge, consisting of (s, a, r, s') transition tuples. Human experience transitions were stored in an additional replay memory during training. We performed Q-learning updates for 15 million frames on minibatches of size 32, composed of 16 samples of human experience and 16 samples of online agent experience. Due to the long training times required for training deep Q-networks and the cumbersome process of generating multiple hours of human experience, we only tested this approach on Montezuma's Revenge.

[3] Code and checkpoints are available here: https://github.com/ionelhosu/atari-human-checkpoint-replay

5.3 Evaluation procedure

In this paper, we used the human starts evaluation metric [17] to test the performance of the agents. The metric consists of using random checkpoints sampled from human experience as starting points for the evaluation of an agent. More specifically, we use a set of 100 checkpoints as human-generated start frames. In order to prove the robustness of our agent, the set of checkpoints used for evaluation are different than the ones that were used for training. This evaluation method averages the score over 100 evaluations of 30 minutes of game time. The value of ε was fixed to 0.05 throughout the evaluation process.

The random agent's scores were obtained using the same evaluation procedure. However the next action to be performed in the environment was sampled from an uniform distribution.

5.4 Quantitative Results

As it can be observed in Table 1, the human checkpoint replay method provides a substantial improvement over a random agent for both games. In Montezuma's Revenge it obtains more than double the points of a random agent. In Private Eye, a random agent is not able to obtain a positive score, due to the multitude of negative rewards present in the game which the agent is not able to avoid. Our HCR DQN agent obtains significantly better results, demonstrating the success of this approach.

Compared to the HCR DQN agent, the human experience replay method provides only slightly better performance over a random agent in Montezuma's Revenge. Due to sparsity of rewards during the game, human experience alone cannot provide enough transitions that lead to positive rewards in order to facilitate learning, although it does provide a slightly better exploration compared to the random agent.

Table 1. Results obtained by our methods, human checkpoint replay (HCR DQN) and human experience replay (HER DQN) on Montezumas Revenge and Private Eye. The results represent raw game scores and were obtained using the human starts evaluation metric.

	Random Agent	**HCR DQN**	HER DQN
Montezuma's Revenge	177.1	**379.1**	218
Private Eye	–41	**1264.4**	N/A

5.5 Qualitative Results

We can draw better insights in the exploration of the agents by taking a closer look on the actions chosen by an agent. For Montezuma's Revenge, the HCR DQN agent is successfully collecting nearby rewards for all start points. For example, in the initial room of level 1, it successfully learns to climb the leftmost ladder in order to get the key. However, the agent still does not learn to avoid monsters and objects that lead to hypothetical negative rewards (such as losing a game life). This is mostly due to the fact that the game does not feature any negative rewards seen as changes in the score. This makes it even more difficult to find a successful exploration policy. While ALE offers this possibility, we did not provide our agents with an additional reward signal which penalizes the agent when it loses a life. It is also important to mention that in the majority of ALE checkpoints generated for training on Montezuma's Revenge there is no reward nearby. As a consequence, this set of checkpoints will continue to provide a challenge for future architectures, as it preserves much of the game's initial difficulty.

In Private Eye, the HCR DQN agent successfully collects most of the nearby rewards, and is seen successfully avoiding objects that lead to negative rewards. The agent does a great job at the latter task, especially as some of the negative rewards the avatar must avoid are moving fast and in an unpredictable manner, making this a difficult task even for human players.

5.6 Discussion

Using human checkpoint replay might be seen as a trade-off for developing general game playing agents. The main objection would be that the agent uses human-generated start points for training the exploration model, which can be seen by some as an "deus ex-machina" intervention for the agent. However, the checkpoint replay merely provides additional starting points and does not offer an understanding of the explored game. It uses the experience of a human player to reach "easier" starting points, but for more difficult games this kind of intervention might be needed.

We should take into consideration that human players also make use of commonsense knowledge for these more difficult games. Using checkpoint replay does not bring any of this prior knowledge directly to the agent, maybe only in an indirect fashion as the human used it to get to that specific game positions.

6 CONCLUSION

In this paper we presented a novel method using deep reinforcement learning, called human checkpoint replay, which was designed for some of the most difficult Atari 2600 games from the Arcade Learning Environment. Our experiments show a substantial improvement compared to all previous learning approaches, as well as over a random player. Our method draws inspiration from curriculum learning and it serves the purpose of compensating for the difficulties of current exploration strategies to find successful control policies in environments with sparse rewards.

As the results show, this method is a promising path of research. We will continue to study other approaches that deal with incentivizing and facilitating exploration in the most difficult games from the Atari platform. We believe that successfully learning control policies in such environments is closely related to the problem of achieving artificial general intelligence as in most real-life situations rewards are not encountered very frequently.

REFERENCES

[1] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato, 'Firm: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements', *The International Journal of Robotics Research*, 0278364913501564, (2013).

[2] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan C. Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu, 'Deep speech 2: End-to-end speech recognition in english and mandarin', *CoRR*, **abs/1512.02595**, (2015).

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling, 'The arcade learning environment: An evaluation platform for general agents', *Journal of Artificial Intelligence Research*, (2012).

[4] Marc G Bellemare, Joel Veness, and Michael Bowling, 'Investigating contingency awareness using atari 2600 games.', in *AAAI*, (2012).

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston, 'Curriculum learning', in *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, (2009).

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, 'Openai gym', *arXiv preprint arXiv:1606.01540*, (2016).

[7] Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu, 'Deep blue', *Artificial intelligence*, **134**(1), 57–83, (2002).

[8] Hado V Hasselt, 'Double q-learning', in *Advances in Neural Information Processing Systems*, pp. 2613–2621, (2010).

[9] Matthew Hausknecht, Piyush Khandelwal, Risto Miikkulainen, and Peter Stone, 'Hyperneat-ggp: A hyperneat-based atari general game player', in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pp. 217–224. ACM, (2012).

[10] Matthew Hausknecht, Joel Lehman, Risto Miikkulainen, and Peter Stone, 'A neuroevolution approach to general atari game playing', *Computational Intelligence and AI in Games, IEEE Transactions on*, **6**(4), 355–366, (2014).

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', *CoRR*, **abs/1512.03385**, (2015).

[12] Erez Karpas and Carmel Domshlak, 'Cost-optimal planning with landmarks.', in *IJCAI*, pp. 1728–1733, (2009).

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 'Imagenet classification with deep convolutional neural networks', in *Advances in neural information processing systems*, pp. 1097–1105, (2012).

[14] Long-Ji Lin, 'Reinforcement learning for robots using neural networks', Technical report, DTIC Document, (1993).

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, 'Playing atari with deep reinforcement learning', *arXiv preprint arXiv:1312.5602*, (2013).

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., 'Human-level control through deep reinforcement learning', *Nature*, **518**(7540), 529–533, (2015).

[17] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al., 'Massively parallel methods for deep reinforcement learning', *arXiv preprint arXiv:1507.04296*, (2015).

[18] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy, 'Deep exploration via bootstrapped dqn', *arXiv preprint arXiv:1602.04621*, (2016).

[19] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver, 'Prioritized experience replay', *arXiv preprint arXiv:1511.05952*, (2015).

[20] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., 'Mastering the game of go with deep neural networks and tree search', *Nature*, **529**(7587), 484–489, (2016).

[21] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 1998.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, 'Going deeper with convolutions', *CoRR*, **abs/1409.4842**, (2014).

[23] Hado Van Hasselt, Arthur Guez, and David Silver, 'Deep reinforcement learning with double q-learning', *arXiv preprint arXiv:1509.06461*, (2015).

[24] Harm Van Seijen and Richard S Sutton, 'Planning by prioritized sweeping with small backups', *arXiv preprint arXiv:1301.2343*, (2013).

[25] Christopher JCH Watkins and Peter Dayan, 'Q-learning', *Machine learning*, **8**(3-4), 279–292, (1992).

FraMoTEC: Modular Task-Environment Construction Framework for Evaluating Adaptive Control Systems

Thröstur Thorarensen[1] **Kristinn R. Thórisson**[1,2] **Jordi Bieger**[1] and **Jóna S. Sigurðardóttir**[2]

Abstract. While evaluation of specialized tools can be restricted to the task they were designed to perform, evaluation of more general abilities and adaptation requires testing across a large range of tasks. To be helpful in the development of general AI systems, tests should not just evaluate performance at a certain point in time, but also facilitate the measurement of knowledge acquisition, cognitive growth, lifelong learning, and transfer learning. No framework as of yet offers easy modular composition and scaling of task-environments for this purpose, where a wide range of tasks with variations can quickly be constructed, administered, and compared. In this paper we present a new framework in development that allows modular construction of physical task-environments for evaluating intelligent control systems. Our proto- task theory on which the framework is built aims for a deeper understanding of tasks in general, with a future goal of providing a theoretical foundation for all resource-bounded real-world tasks. The tasks discussed here that can currently be constructed in the framework are rooted in physics, allowing us to analyze the performance of control systems in terms of expended time and energy.

1 INTRODUCTION

To properly assess progress in scientific research, appropriate evaluation methods must be used. For artificial intelligence (AI) we have task-specific benchmarks (e.g. MNIST [21]) for specialized systems on one end of the spectrum and—proposed yet controversial—tests for more general human-level AI (e.g. the Turing test [40]) on the other end. Little is available on the middle ground: for systems that aspire towards generality, but are not quite close to human-level intelligence yet. A major goal of AI research is to create increasingly powerful systems, in terms of autonomy and ability to address a range of tasks in a variety of environments. To evaluate the general ability and intelligence of such systems, we need to test them on a wide range of realistic, unfamiliar task-environments [17]. To cover the entire spectrum of AI systems, we want to be able to analyze, compare and adapt the task-environments that we use [36].

A prerequisite to evaluating systems is the construction of task-environments. It has been suggested that successful regulation or control implies that a sufficiently similar model must have been built (implicitly or explicitly) [9, 32]. It is important for AI to understand how the construction of models of task-environment works: for evaluation, for the model-building (learning) and decision-making that goes on in the "mind" of a successful controller, and for the design process where match between agent and task must be considered. Tasks truly are at the core of AI, and in another paper (under review) we argued for the importance of a "task theory" for AI [37]. Other (engineering) fields often have a strong understanding of the tasks in their domains that allows them to methodically manipulate parameters of known importance in order to systematically and comprehensively evaluate system designs. A task theory for AI should provide appropriate formalization and classification of tasks, environments, and their parameters, enabling more rigorous ways of measuring, comparing, and evaluating intelligent behavior. Analysis and (de)construction capabilities could furthermore help any controller make more informed decisions about what (sub)tasks to pursue, and help teachers manipulate or select environments to bring about optimal learning conditions for a student [4].

Here we present an early implementation of a framework for the construction of modular task-environments for AI evaluation, based on the initial draft of a task theory. The modular nature of the construction makes it easy to combine elementary building blocks into composite task-environments with the desired complexity and other properties. It allows us not only to make simple variants on existing task-environments to test generalization ability and transfer learning, but also to measure (learning) progress by scaling environments up or down (e.g. by adding/removing building blocks). The framework is aimed at general systems that aspire to perform real tasks in the real world. Such systems have bounded resources that they must adequately manage, so we take a special interest in the amount of time and energy they use to perform assigned tasks. Since we are mainly interested in assessing cognitive abilities, the agent's body is defined as part of the task-environment so that we can evaluate the performance of the agent's controller (i.e. its "mind").

2 RELATED WORK

Since the inception of the field of AI, the question of how to evaluate intelligence has puzzled researchers [40]. Since then a lot of work in the area has been done [22, 17, 25]. Most AI systems are developed for a specific application, and their performance is easily quantifiable in domain-specific terms. Evaluation of more general-purpose algorithms and systems has always been more difficult. A common strategy is to define benchmark problems (e.g. MNIST written character recognition [21] or ImageNet object detection challenges [10]), AI-domain-restricted contests (e.g. the Hutter compression prize [19], planning competitions [8] or the DARPA Grand Challenge [39]) or iconic challenges (e.g. beating humans at chess [7] or Go [33]) in the hope that performance on these manually selected problems will be indicative of a more general kind of intelligence. These methods are good at evaluating progress in narrow domains of AI, where they

[1] Center for Analysis and Design of Intelligent Agents, School of Computer Science, Reykjavik University, Iceland. email: {throstur11,thorisson,jordi13}@ru.is
[2] Icelandic Institute for Intelligent Machines, Reykjavik, Iceland. email: jona@iiim.is

encourage innovation and competition, while also—unfortunately—encouraging specialized approaches for overfitting on the evaluation measure.

Many methods aimed at more general intelligence exist as well. Many researchers have turned to theories of human psychology and psychometrics to evaluate AI systems, resulting in human-centric tests (e.g. the Turing Test [40], the Lovelace Tests [5, 27], the Toy Box Problem [20], the Piaget-MacGuyver Room [6] and AGI Preschool [14, 15]; see also the latest special issue of AI Magazine [25]). These tests tend to either be very subjective, very human-centric or only provide a roughly binary judgment about whether the system under test is or is not (close to) human-level intelligent. They are again applicable to only a narrow slice of AI systems, which in many cases don't exist yet (i.e. we have no human-level AI that can genuinely pass most of these tests).

What we need is a way of evaluating AI systems that aspire towards some level of generality, but are not quite there yet [36]. Hernández-Orallo argued that in order to assess general intelligence, that the assessment should cover the testing of a range of abilities required for a range of tasks [17]. What is needed then is a battery of tasks that can be used to evaluate the cognitive abilities of a system. Ideally, this battery of tasks should be suitable to the system we want to test, but still comparable to tasks that are used on other systems.

The importance of using a range of tasks that are unknown to AI system designers is widely—although not nearly unanimously—recognized. For instance, the 2013 Reinforcement Learning Competition [42, 1] included a "polyathlon" task in which learning systems were presented with a series of abstract but related problems. We see the same in other competitions: the General Game Playing competition [12] presents contestants with the description of a finite synchronous game only moments before it needs to be played, and the General Video Game AI competition [24] does the same with video games. These competitions are very interesting, but their domains are still fairly restricted, the adversarial nature makes progress evaluation between years tricky, and the "tasks" each need to be carefully constructed by hand.

To really be able to evaluate a wide range of AI systems well, it is necessary that tasks—and variants of those tasks—can easily be constructed or preferably generated. Hernández-Orallo advocates this approach, but only covers discrete and deterministic environments [16, 18]. Legg & Veness developed an "Algorithmic IQ" test that attempts to approximate a measure of universal intelligence by generating random environments [23]. Unfortunately these methods cannot easily generate complex structured environments and are opaque to analysis and human understanding. Our own prior work on the MERLIN tool (for Multi-objective Environments for Reinforcement LearnINg) [11] followed in the footsteps of other Markov Decision Process generators like PROCON [2] and GARNET [3]. While Merlin does support using continuous state and action spaces and somewhat tunable environment generation, it fails to meet most of the requirements below.

In [36] we listed a number of requirements for the comprehensive evaluation of artificial learning systems. An ideal framework ought to cover the complete range of AI systems—from very simple to very advanced. Performance of various systems on different tasks should be comparable, so as to differentiate between different systems or measure progress of a single system. Such a framework would not only facilitate evaluation of the performance of current and future AI systems, but go beyond it by allowing evaluation of knowledge acquisition, cognitive growth, lifelong learning, and transfer learning. Most importantly, it should offer easy construction of task-environments and variants, the ability to procedurally generate task-environments with specific features, and facilitation of analysis in terms of parameters of interest, including task complexity, similarity and observability. Easy construction includes the ability to compose, decompose, scale and tune environments in terms of parameters like determinism, ergodicity, continuousness, (a)synchronicity, dynamism, observability, controllability, simultaneous/sequential causal chains, number of agents, periodicity and repeatability.

The framework we present here is a prototype aimed towards the requirements outlined above. Section 7 will elaborate on this more, as we evaluate our success so far.

3 TASK THEORY

The concept of *task* is at the core of artificial intelligence (AI): tasks are used in system evaluation, training/education and decision making. Tasks can vary from the classification of an image, to the clustering of data points, and to the control of a (physical or virtual) body to cause change in an environment over time. It is the latter kind of task that we are primarily concerned with here.

Most AI systems are designed to perform a *specific kind* of task, and most systems require a set of concrete tasks to train on in order to learn to later perform similar tasks in production. This requires the collection or construction of appropriate task examples.

Systems that aspire to a more general kind of intelligence aim to tackle a wide range of tasks that are largely unknown at design time. Upon deployment these systems are intended to not rely on their designer to decompose their future tasks into component parts and elementary actions – they will need to choose among different decompositions and (sub)tasks themselves, in terms of both priority (benefits) and feasibility (costs). Evaluation of more general cognitive abilities and intelligence can not simply be done by measuring performance on a single target task: we could just develop a specialized system for that[3]. Rather, we need a battery of tasks that can be modified to grow with the systems under test and facilitate the measurement of knowledge acquisition, cognitive growth, lifelong learning, and transfer learning.

While we don't have fully general systems yet, different systems will need to be evaluated on different task batteries, and we need the flexibility to tailor those to the systems under test and the ability to compare performance on various tasks in order to compare different AI systems. Yet in most cases tasks are selected ad hoc, on a case-by-case basis without a deep understanding of their fundamental properties or how different tasks relate to each other. We have argued elsewhere for the importance of a "task theory" for AI [37]. Such a theory should cover all aspects of tasks and the environments that they must be performed in, and cover:

1. *Comparison* of similar and dissimilar tasks.
2. *Abstraction* and *reification* of (composite) tasks and task elements.
3. Estimation of time, energy, cost of errors, and other resource requirements (and yields) for *task completion*.
4. Characterization of task complexity in terms of (emergent) quantitative measures like *observability*, *feedback latency*, form and nature of *information/instruction* provided to a performer, etc.
5. Decomposition of tasks into subtasks and their atomic elements.

[3] Unless the single target task is AI-complete (c.f. the Turing test and similar tests), but these are typically only applicable to a very narrow range of intelligence (i.e. humanlike intelligence), and no current systems can pass these tests.

6. Construction of new tasks based on combination, variation and specifications.

To accomplish all of this we need some way to formalize task-environments. Here we only give an overview of our initial attempt; for more detail see [37].

At the highest level, we use a tuple of task and environment—similar to Wooldridge's $\langle Env, \Psi \rangle$ [43], where Ψ represents the criteria by which success will be judged. As a first approximation a *task* may be formulated as an *assigned goal*, with appropriate constraints on time and energy. Wooldridge [43] defines two kinds of goals, what might be called *achievement goals* ("Ensure $X \approx G_X$ before time $t \geq 10$") and *maintenance goals* ("Ensure $X \approx G_X$ between time $t = 0$ and $t = 10$).[4] By introducing time and energy into the success criteria, this disparity is removed: Since any achieved goal state must be held for some non-zero duration (at the very minimum to be measured as having been achieved) an achievement goal is simply one where the goal state may be held for a short period of time (relative to the time it takes to perform the task which it is part of) while a maintenance goal is held for relatively longer periods of time. The highest attainable precision of a goal state is defined by the laws of physics and the resolution of sensors and actuators. Performing a task in the real world requires time, energy, and possibly other resources such as money, materials, or manpower. Omitting these variables from the task model is tantamount to making the untenable assumption that these resources are infinite [41]. Including them results in a unified representation of goals where temporal constraints on the goal state are provided.

Any action, perception and deliberation in the real world takes up at least some time and energy. Limitations on these resources are essentially the raison d'être of intelligence [35]—unbounded hypothetical systems will randomly stumble upon a solution to anything as time approaches infinity. Estimation of time, energy and other resource requirements (and yields) for task completion can be used to design effective and efficient agent bodies, judge an agent based on comparative performance, and make a cost-benefit analysis for deciding what (sub)tasks to pursue.

For the environment part of our formalization we take inspiration from Saitta & Zucker [30]. The environment description contains variables describing objects in the world, as well as transition functions to describe the dynamics. Environments are considered perspectives on the world, and can be nested within each other if desirable, resulting in a certain amount of modularity. Since we are mainly interested in creating systems with advanced cognitive abilities, we define sensors and actuators simply by listing observable and controllable variables. This essentially places the agent's body inside the environment. From an evaluation perspective, this allows us to focus on the agent's *controller* (mind). From a task analysis point of view this allows to make statements about physical limits and feasibility without needing to consider the unknown cognitive abilities of a controller.

4 FRAMEWORK

Our Framework for Modular Task-Environment Construction "FraMoTEC" enables the construction and simulation of task-environments in a modular way. An early prototype has been implemented in Python in an object-oriented manner, using a layered composition of small building blocks to describe entire environments.[5] To aid in the explanation of our framework and its various components, we use the following running example of a task-environment:

Example 1 (Car Race)**.** A one-dimensional race is our simplest version. The agent must move a car across the finish line N meters away from the starting position. The controller determines the rate of energy expenditure, which results in higher or lower acceleration (pressing the gas pedal more increases the flow of fuel to the engine and through that the amount of energy that is converted in order to move the vehicle). Naturally the race must be finished using as little time as possible and using an amount of energy that doesn't exceed what is available from the gas tank.

Example 2 (Car Parking)**.** Example 1 can straightforwardly be converted to a parking task if we require the car to end up between two points (rather than across a finish line), or extended by e.g. adding a second dimension or adding/removing friction and air resistance.

4.1 Components

Constructing task-environments requires using the building blocks provided by the framework. These building blocks are designed to be as basic as possible to allow for a wide variety of behaviors to emerge from the different combinations of organization of the blocks. Most of the organizational complexity of the resulting tasks emerges from the various combinations of objects with custom transitions. The following components have been incorporated: objects, transitions, systems, goals, motors and sensors. When all building blocks come together they form what is finally called the "model"—i.e. the complete representation of the task-environment (and by extension, the natural system that the model represents).

Objects Objects are used to describe the "things" in a task-environment model, such as the `car` in the race example. The framework implements basic one-dimensional kinematics individually for each object. Objects have a main `value` x (in the example corresponding to the car's position) as well as physical properties like `velocity` v, `mass` m, `friction` μ_k and might even contain values for gravitational acceleration at some `angle` θ. This allows the object to naturally transition (as will be explained below): the new `velocity` is computed based on the current `velocity` and the input power P (and direction) from any affectors, which is then used to update the main `value`, as shown by these physics equations:

$$\begin{aligned}
F_{\text{input}} &= \frac{P}{v} \\
F_{\text{gravity}} &= -mg \cdot \sin\theta \\
F_{\text{friction}} &= -\operatorname{sgn} v \cdot \mu_k mg \cdot \cos\theta \\
F_{\text{total}} &= F_{\text{input}} + F_{\text{gravity}} + F_{\text{friction}} \\
v &\leftarrow v + \delta t \cdot \frac{F_{\text{total}}}{m} \\
x &\leftarrow x + \delta t \cdot v
\end{aligned}$$

where g is the gravitational constant and δt is the (theoretically infinitesimal) time over which the framework calculates each change.

Although the framework does not currently implement other object behavior, we envision extending the framework as experience with it accumulates.

[4] We use approximate rather than precise equivalence between X and its goal value G_X because we intend for our theory to describe real-world task-environments, which always must come with error bounds.

[5] The FraMoTEC code is available at https://github.com/ThrosturX/task-env-model.

Transitions Transitions or transition functions are used to change the (values of) objects in the task-environment. Transitions come in two forms: the *natural* form and the *designed* form. Natural transitions describe the natural change of the objects and systems in the task-environment. They are provided by the framework and executed automatically during simulation unless this is explicitly prevented. Transitions that are specified by the task-environment designer expand upon the natural behavior of an environment by adding custom logic to it without requiring the framework to be extended specifically. We could for example implement a transition that causes the car in our race example to lose mass as its energy is depleted: `t_mass: car.mass` $\leftarrow$ `1200 + car.energy / 46000`.

Motors Motors can be considered "effectors" or "actuators" of the controller, which it can directly interact with to affect the environment, to achieve goals and perform tasks. The controller sets the rate at which energy is transferred to each such motor (we refer to this energy transfer rate as "power"). When connected to an object (as they typically are), this energy is converted into a force that affects an object's current `velocity`. Motors can be placed in systems of objects with custom transitions to create new behavior. For instance, to add more realistic steering controls, instead of letting the controller use independent motors to affect the `x` and `y` position directly, we could add motorized `orientation` and `speed` objects, plus these transitions:

- `x.velocity` $\leftarrow$ `speed.velocity`$\cdot\cos($`orientation.value`$)$
- `y.velocity` $\leftarrow$ `speed.velocity`$\cdot\sin($`orientation.value`$)$

Sensors Standardized *access* to objects' `value`s can be provided via sensors. A sensor reads an object's `value`, optionally applying some distortion due to noise or limited resolution. Sensors can also read other sensors, allowing designers to combine observations or apply multiple layers of distortions.

Systems Systems facilitate composition immensely by acting as a container for objects, transitions, sensors, motors and other elements. The natural transition of a system is to apply all transitions within. Systems can be used to create a hierarchy of larger building blocks that can easily be reused and rearranged. For instance, we could create a system to encapsulate the above object `car` and transition `t_mass` so that more cars whose mass depends on the contents of their fuel tank can easily be made. Or, when we define a car that can move in two dimensions, we need separate objects for the position in each dimension. We could make a system with two objects—for `x_position` and `y_position`—and motors to control each dimension separately (this would perhaps be more reminiscent of a helicopter) or motors for controlling speed and angle of the wheels. One kind of "car" could easily replace another kind in a larger system, without affecting everything else, making it easy to create slight variations. "Inaccessible" objects or other constructs can also be created, whose value is inaccessible directly via sensors. This facilitates the theoretical creation of systems with hidden states.

Goals Goals are used to define tasks that the controller must perform. A goal specifies a target object X along with a goal value G_X, tolerance ϵ and time restrictions. Tolerance values should be used because all measurements are constrained by real-world resolution. Time restrictions should allow the user to specify before and after which (absolute and relative) times the target value needs to be in the goal range. Goals can also depend on other goals to be satisfied (i.e. the goal can require other goals to be met before it can consider itself satisfied). This allows users to easily define composite tasks by sequencing goals. Once a goal has been satisfied, it is forever considered satisfied unless it is reset, in which case both the goal itself and any goals it was a prerequisite will be reset recursively. This allows the state of the *task* to be evaluated based on the number of achieved goals without regarding the *environment's* current state. In the car race example we might define a goal that says $G_X - \epsilon \leq$ `car.value` $\leq G_X + \epsilon \wedge t \leq 10$. Since the task is accomplished as soon as the goal becomes satisfied for the first time, `tolerance` could be set to zero. For the parking task, we might add a goal that says $0 - \epsilon \leq$ `car.velocity` $\leq 0 + \epsilon$ and add it as a prerequisite to the other goal to demand that the car is stopped at the end.

Task-Environment Model Finally, a model of an entire task-environment is created that can be run by the framework. The task part consists of a set of goals plus additional restrictions on time and energy. The environment is a system that contains all relevant objects, transitions, sensors and motors.

5 Task Construction

Tasks are constructed modularly using the smallest possible constructs, as building blocks with peewee granularity provide the most flexibility [38]. The simplest conceivable task-environment is essentially specified in our car race example: the environment is a single system containing one object with an associated motor and sensor, and the task simply specifies a single goal value for the only object along with some restrictions on time and energy. We can construct more complicated tasks by adding more objects, sensors and motors to the environment. We can enforce systems' behavior as desired by implementing appropriate transitions. We can for example create a system in which objects X and Y move independently except if $Y < 20$ by implementing a transition that checks if $Y < 20$, locking X in place (freezes it's state) if so, and unlocking it otherwise: $\textit{transition}_{\textit{lock-x}}$: `X.locked` $\leftarrow$ `Y.value` ≥ 20.

The framework has some built-in randomization options, allowing designers to set ranges of options for the values of objects and goals. This allows us to easily generate a set of different but highly similar concrete tasks that can be used to train or evaluate a system on the general idea of a task family, rather than just having it memorize the specifics of a single task.

5.1 Simulation

In order to evaluate or train intelligent controllers, it is important that the constructed task-environments can be executed or simulated. The designers of task-environments ultimately produce formalizable models—this is a natural implication of the framework building on simple, simulable causal processes (the building blocks and their interaction). A simulation of the model becomes a simulation of the natural system that the model represents, transmuting Church's thesis into an assertion (all systems that can be modeled by the framework are simulable) [28].

In order to simulate a task-environment, its model needs to be connected to a controller. Since different controllers have different requirements, this is largely left up to the user. FraMoTEC is implemented in the Python programming language and provides a simple API on the task-environment model. The typical method of usage would be to construct and instantiate a task-environment model as well as a control system—here we will use the example of a SARSA reinforcement learner. The user is free to access all aspects of the task-environment and should use this access to define the interaction with the controller in any way that they like. For instance, some implementations of table-based SARSA systems may need to be instantiated with knowledge of the range of possible actions and observations.

The task-environment model provides a `tick` method that takes `delta_time` as an argument. Whenever this method is called, the framework will run the simulation for `delta_time` seconds. When the method returns, the user can easily obtain the values from sensors to pass on to the controller. At this point it is also possible to e.g. compute a reward based on values from the sensors or other information that the user can access about the current state of task and environment. It should be noted that the framework itself does not yet support explicit rewards.[6] Not every control system requires explicit or incremental rewards; for more powerful learners rewards for ought to be intrinsic [34, 31, 13, 26]. Having said that, it is trivial to implement rewards via a special kind of sensor designated as a reward channel, and to possibly couple this with the state of goal achievement, which is of course tracked within the system. The user could then pass on the actions of the controller to the motors of the task-environment before calling `tick` again. This fully synchronous mode of interaction is required by most reinforcement learners, but FraMoTEC could also run simulations asynchronously if the controller supports it.

The simulation component of the framework would ideally be truly continuous, but the nature of the Von Neumann architecture encourages stepwise integration. `delta_time` can be viewed as the time resolution of the controller. The task-environment model has its own time resolution `dt`. As such, every simulation step regardless of length should optimally ensure that:

- For all systems: naturally transition for `dt` seconds—*recall that any system's natural transition fires all transitions within*.
- For all objects: naturally transition for `dt` seconds
- Goals should be asserted to evaluate whether success (or failure) conditions have been met
- The time passed during the frame must be recorded and added to an accumulator
- The energy used by any motor during that time frame should be recorded and added to an accumulator
- Current time and energy usage should be compared with time and energy limits

5.2 Analysis

In the current prototype implementation FraMoTEC offers limited functionality for analyzing task-environments and the performance of controllers. For a given task, the framework can produce a time-energy tradeoff plot (Pareto curve) that shows the minimal amount of energy that is required to finish the task in a given amount of seconds (or alternatively: the fastest the task can be completed given a certain amount of expended energy).

As previously established, time and energy usage are key metrics to consider when evaluating the performance of controllers in a given set of task-environments. It goes without saying that a controller that spends 2 minutes and 20 KJ of energy to solve a specific task-environment is worse at *completing the task* than a controller that spends 30 seconds and 700 J in that same task-environment. Maybe the first controller spent more time actually learning about the environment, in which case it might be much better suited for a set of similar task-environments than the second controller.

Naturally, we can continuously measure the time and energy expenditure of an controller to quantify the total amount of time and energy required to come up with a solution to some task. In this sense we are not evaluating a controller's ability, but its ability to improve some ability (i.e. the controller's ability to learn). We can further extend both these evaluation methods to a set of tasks in lieu of a single task, allowing for a more comprehensive evaluation and comparison of all kinds of controller.

After simulation, successful attempts by the controller that resulted in completing the task can be added to the graph to compare time and energy usage compared to each other and the optimal Pareto curve. Different runs are color-coded according to the time at which they occurred (earlier attempts are lighter, while later ones are darker), which shows a controller's (hopefully improving) behavior over time.

6 USE CASES

In this section we will showcase some simple use cases of the system. Since this paper is not about advanced control systems themselves, we will use a simple SARSA reinforcement learner [29] and some domain-specific control systems to illustrate the capabilities of the framework in a simple way.

6.1 Learning Agent

An agent was implemented with an underlying SARSA reinforcement learning algorithm. The state exposed to the agent was an *n-tuple* of all sensor readings along with the velocity of one of the objects in the model. A scoring function was implemented to determine rewards[7].

Reinforcement learners generally learn slower as the $state{\cdot}action$ space increases, therefore the agent enumerates the available actions as the setting of a single motor at one of three power levels: (i) 0 (ii) P_{max} and (iii) $-P_{max}$. We experimented with an agent that included the settings (iv) $\frac{P_{max}}{2}$ and (v) $-\frac{P_{max}}{2}$, but we found that these settings unnecessarily crippled the agent and removed them. The agent implements a method `perform(self, dt)` that creates an *experience* for the agent by: (a) setting the current state (b) selecting and executing the *reward-maximizing action* (c) ticking the simulation by *dt* seconds (d) rewarding the learner based on the value of the scoring function and the new state. This method is called repeatedly in the evaluation, see Section 6.2.1.

6.2 Task-Environments

The agent was introduced to two similar environments. The first environment had the goal of moving the `position` object into `goal_position` with a tolerance of 5, with 5000J and 60 seconds as the maximum expendable time and energy (essentially, the 1D car race example):

[6] Rewards are appropriately seen as part of the information/training materials for a task, not as part of the task proper (although one may argue that a task will change drastically, perhaps fundamentally, depending on what kind of information is given about it up front and during its learning/performance).

[7] Implemented as $\left(-\sum_{i=0}^{N} |s_{object_i} - s_{goal_i}| - \epsilon_i\right)$ where s represents the position (value) of either the object or the goal associated with a `Goal` in the task-environment's `solution`.

- One object: `position`
- One fully reversible motor affecting `position` with 200W maximum input
- One sensor for `position`
- Goal: `position` within `goal_position` ± `goal_epsilon`
- Max time and energy: 60 s, 5000 J

The second environment expanded upon this environment, requiring a "plotter" to be activated when the position is correct—both goals needed to be satisfied to consider the task solved. An additional transition in the second environment locked the position while the plotter was activated.

- Two objects: `position` and `plot_it`
- One fully reversible motor affecting `position` with 200 W maximum input
- One non-reversible motor affecting `plot_it` with 5 W maximum output[8]
- One sensor for each object
- **New transition function**: If `plot_it` >0.5: `position` is locked, otherwise it is unlocked.
- Goal prerequisite: `position` between `goal_position` ± `goal_epsilon`
- Goal: `plot_it` is 1 ± 0.1
- Max time and energy: 60 s, 10000 J

The second task-environment increases the task difficulty when compared to 1D car racing by adding a new object (complete with sensor and motor), changing the behavior (with the transition function that locks the `position` object) and by expanding on the original goal.

6.2.1 *Evaluation*

First, the agent is tasked with solving some training environments which are copies of the target environment, except with a more favorable starting position. The training environments gradually get more difficult by increasing the distance between the starting position and the goal. Once this training is complete, the agent gets 200 chances to satisfy the goal(s) in each task-environment. The data is visualized using the methods described in Section 5.2. Figure 1 shows the results for the 1D car race task-environment. Figure 2 shows the results for the 1D locking plotter task-environment. Note that the agent continues to learn by creating *experiences* during the evaluation (i.e. learning is not "switched off"). The evaluation works as follows:

- While the task is not solved:
  1. If the task has been *failed*, stop
  2. Invoke the agent's `perform` method (with dt set to $0.25s$, see Section 6.1).
- Finally, report time and energy usage (and indicate if the task failed).

Note on reading the plots: The blue line represents the energy required to complete the task at each time. The red line represents the maximum amount of expendable energy due to motor power limitations. The dots represent data points for the evaluations, with lighter colored (greener) data points representing earlier runs and darker colored (redder) data points representing later runs.

[8] You can think of a solenoid with an off button.

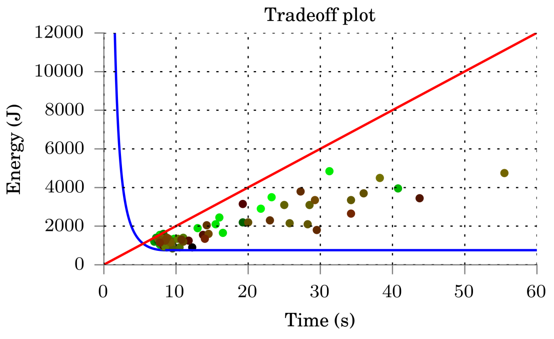


Figure 1. Resulting plot for 200 evaluations in the 1D car race environment.

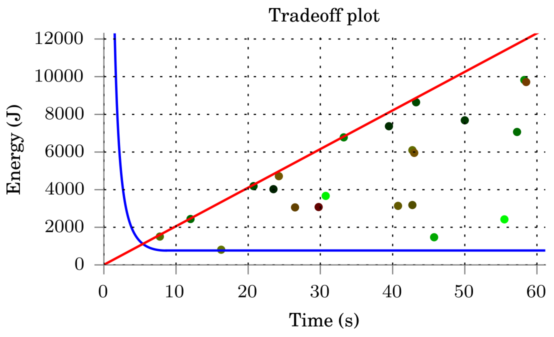


Figure 2. Resulting plot for 200 evaluations in the 1D locking plotter environment.

6.3 Agent Comparison

In order to demonstrate how different agents can be compared just as different environments can be compared, a custom agent implementation was compared with the SARSA implementation in the 1D locking plotter environment. The custom agent roughly implements the following algorithm in the `perform` method:

- Compute `distance` between `position` and the corresponding goal
  - If the distance is small enough, deactivate the `position` motor and activate the `plot_it` motor.
  - If the distance is positive, maximum power to the `position` motor and deactivate the `plot_it` motor.
  - If the distance is negative, maximum negative power to the `position` motor and deactivate the `plot_it` motor.
- Tick the simulation by dt seconds

It should be obvious that the above algorithm is specifically tailored to outperform the SARSA agent, as it includes domain knowledge which the SARSA agent would need to come up with on its own. Figure 3 indeed shows that this controller performs much better and more constantly, but also that it's not improving over time.

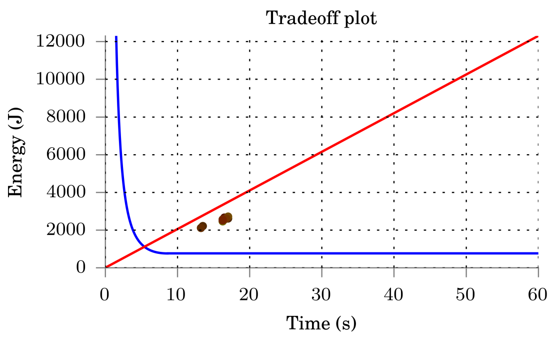


Figure 3. Resulting plot for 200 evaluations in the 1D locking plotter environment using an agent with a domain-specific implementation.

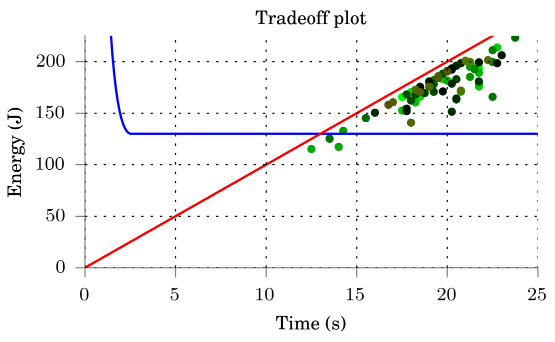


Figure 4. Resulting plot for 100 evaluations in a generic 10-dimensional task-environment.

6.4 N-Dimensional Task Comparison

6.4.1 Task-Environments

A generic N-dimensional task-environment generator is included in the samples as `sample_N_task`. The generator returns a task-environment with N objects and associated sensors with a default starting position of 3 ± 2 and a goal of reaching 10 ± 0.5. There are two systems: (i) a control system which contains two objects with associated motors and sensors and a transition function that sets the power level of some hidden motor to some value depending on the values of the objects in the control system (ii) a hidden motor system which ensures that activating the hidden motors for each of the N variables results in that power usage being counted

The control system includes the motors that the agent should have direct access to. The main power motor determines how much power is input into the hidden motors while the selection motor determines which hidden motor is activated.

6.4.2 Controller

A simple controller was created to solve the class of task-environments described in the previous section. The algorithm is quite simple, the below should demonstrate the agents `perform` method:

- Activate the main power motor
- Determine the object that is furthest from the goal, call it `min_o`
- Determine the direction of power required to enable `min_o`'s affector
- Activate the selection motor in the appropriate direction
- Tick the simulation by *dt* seconds

6.4.3 Results

Two variations of the N-dimensional task were attempted, one with 10 dimensions and one with 20 (Figures 4 and 5). It should not come as a surprise that the task-environment with fewer dimensions was solved in less time, with less energy. However, the difference was not double, as one might be inclined to suspect when doubling the size of the environment. This gives us an indication about the agent's ability to scale with environments (but could also give some indication of how well-formed the environment itself is). Since we know everything there is to know about the environment, we can assert that the agent seems to scale well from 10 to 20 dimensions.

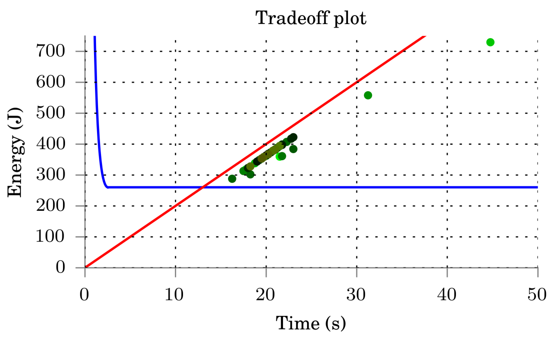


Figure 5. Resulting plot for 100 evaluations in a generic 20-dimensional task-environment.

7 CONCLUSION & FUTURE WORK

In this paper we have presented our ideas for and early prototype implementation of a framework for modular task-environment construction (FraMoTEC). FraMoTEC aims to facilitate the evaluation of intelligent control systems across the entire spectrum of AI sophistication on practical tasks. The framework is intimately intertwined with an equally early-stage "task theory" that is intended to deepen our understanding of fundamental properties of various types of task-environments and how they relate to each other. Such an understanding would help us compare control systems against each other and earlier versions in order to measure progress, learning and growth.

A major goal was to lay the groundwork for an evaluation tool that meets the requirements that we outlined in an earlier paper [36]: facilitation of easy construction, procedural generation and in-depth analysis. We believe that the framework does indeed make steps in the right direction. Current analysis capabilities of the framework are very limited, but already provide for instance some rudimentary understanding of tradeoffs between time and energy, and measuring a learning system's performance increase over time. One major piece of future work is to further develop task theory so that we can make predictions about the effects of combining tasks and environments: e.g. when we add a dimension or additional goal, how does that affect minimum time and energy requirements?

Procedural generation of task-environments is precursory at this point in time. The framework allows users to set ranges of acceptable options for initial values of objects and goals instead of concrete values. Slightly different concrete task-environments can then be instantiated automatically by the framework. However, the modular nature of tasks and environments should make it relatively easy to add functionality for e.g. adding dimensions or sequencing goals.

This modularity also allows for easy construction, composition, decomposition and scaling of task-environments. Adding or removing objects or (sub)systems, as well as combining goals and tasks in various ways, allows us to make simple task-environments (slightly) more complex and vice versa; thereby allowing our battery of tasks to grow with the AI system under test. As the name suggests FraMoTEC is primarily a framework for task-environment *construction* and this is where it shines, even though much work remains to be done.

In [36] we listed a number of properties of task-environments that a framework should 1) support and 2) ideally let the user tune:

1. **Determinism** Both full determinism and stochasticity must be supported. The framework provides the option of partial stochasticity out-of-the-box, such as in the creation of objects (start values can be randomized), goals, sensor readings, and designed transitions.
2. **Ergodicity** Ergodicity controls the degree to which the agent can undo things and get second chances. The framework imposes no restrictions on this other than a fundamental rule: Expended time and energy cannot be un-expended. If the agent spends time or energy doing the wrong thing, that time and energy will still have been spent and the task-environment needs to be reset in order to give the agent a second chance with regard to the energy and time expenditure. Task-environment designers have full control over what states are reachable.
3. **Controllable Continuity** This point notes that it is crucial to allow continuous variables, and that the degree to which continuity is approximated should be changeable for any variable. All objects in the framework contain continuous variables, discretized only by floating-point inaccuracies by default. It is possible use sensors to further discretize (or distort) any accessible variables. It is also possible to tweak the time resolution of the simulation.
4. **Asynchronicity** Any action in the task-environment should be able to operate on arbitrary time scales and interact at any time. This must currently be done manually, and we aim to provide a more user-friendly solution in the future.
5. **Dynamism** The framework gives the user full control over how static or dynamic environments are. Natural transitions of objects can provide some limited dynamism, but controllers can be given a static experience by clever sampling. Most dynamics will come from designed transitions created by the framework user.
6. **Observability** The observability of task-environments is determined by the interface between the environment and the controller interacting with it. Sensors are the primary control for observability in the framework. Sensors can be tuned to tune the observability of a task-environment by distorting the value and/or discretizing it to a user-specified resolution.
7. **Controllability** Controllability is the control that the agent can exercise over the environment to achieve its goals. The controllability of the task-environment is controlled with the exposure of motors to the controller. By modifying motor properties and interactions between motors (specifically in custom transition functions), the controllability of a task-environment can be tuned.
8. **Multiple Parallel Causal Chains** Co-dependency in objectives can be programmed into designed task-environments without hassle. The framework does not place any restrictions on causal chains with a single exception that circular-references are currently not supported (two goals may not mutually depend on each other, one must depend on the other first).
9. **Number of Agents** The framework does not restrict the number of agents nor what interactions can take place. Even if multiple agents have access to the same motors, the framework regards the most recent setting to be the current setting. However, interactions are currently mostly defined by the user. We hope to provide more user-friendly support in the future, which will go hand-in-hand with implementing a better method for asynchronicity.
10. **Periodicity** The framework does not specifically handle periodicity, cycles or recurrent events. The user must implement this with designed transitions if they need this.
11. **Repeatability** By using the same random seed, repeatability can be guaranteed in most circumstances. However, agents and sensors must use their own random number generators (and seeds) to avoid tampering with task-environment repeatability.

While a lot of work remains to be done, we believe that this framework will be able to eventually fulfill the requirements we outlined and significantly contribute to the field of AI evaluation, task theory, and by proxy: AI itself.

ACKNOWLEDGEMENTS

This work was sponsored by the School of Computer Science at Reykjavik University, and a Centers of Excellence Grant (IIIM) from the Science & Technology Policy Council of Iceland.

REFERENCES

[1] ICML workshop on the reinforcement learning competition, 2013.
[2] T. W. Archibald, K. I. M. McKinnon, and L. C. Thomas, 'On the generation of markov decision processes', *Journal of the Operational Research Society*, 354–361, (1995).
[3] Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee, 'Natural actor–critic algorithms', *Automatica*, **45**(11), 2471–2482, (2009).
[4] Jordi Bieger, Kristinn R. Thórisson, and Deon Garrett, 'Raising AI: Tutoring Matters', in *Proceedings of AGI-14*, pp. 1–10, Quebec City, Canada, (2014). Springer.
[5] Selmer Bringsjord, Paul Bello, and David Ferrucci, 'Creativity, the Turing test, and the (better) Lovelace test', *Minds and Machines*, **11**, 3–27, (2001).
[6] Selmer Bringsjord and John Licato, 'Psychometric Artificial General Intelligence: The Piaget-MacGuyver Room', in *Theoretical Foundations of Artificial General Intelligence*, 25–48, Springer, (2012).
[7] Murray Campbell, A. Joseph Hoane Jr., and Feng-hsiung Hsu, 'Deep Blue', *Artificial Intelligence*, **134**(1–2), 57–83, (2002).
[8] Amanda Coles, Andrew Coles, Angel García Olaya, Sergio Jiménez, Carlos Linares López, Scott Sanner, and Sungwook Yoon, 'A survey of the seventh international planning competition', *AI Magazine*, **33**(1), 83–88, (2012).
[9] Roger C. Conant and W. Ross Ashby, 'Every good regulator of a system must be a model of that system†', *International Journal of Systems Science*, **1**(2), 89–97, (1970).
[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, 'Imagenet: A large-scale hierarchical image database', in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, (2009).
[11] Deon Garrett, Jordi Bieger, and Kristinn R. Thórisson, 'Tunable and Generic Problem Instance Generation for Multi-objective Reinforcement Learning', in *Proceedings of the IEEE Symposium Series on Computational Intelligence 2014*, Orlando, Florida, (2014). IEEE.

[12] Michael Genesereth and Michael Thielscher, 'General game playing', *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **8**(2), 1–229, (2014).

[13] Olivier L. Georgeon, James B. Marshall, and Simon Gay, 'Interactional motivation in artificial systems: between extrinsic and intrinsic motivation', in *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pp. 1–2. IEEE, (2012).

[14] Ben Goertzel and Stephan Vladimir Bugaj, 'AGI Preschool: a framework for evaluating early-stage human-like AGIs', in *Proceedings of AGI-09*, pp. 31–36, (2009).

[15] Ben Goertzel, Cassio Pennachin, and Nil Geisweiller, 'AGI Preschool', in *Engineering General Intelligence, Part 1*, number 5 in Atlantis Thinking Machines, 337–354, Atlantis Press, (2014).

[16] José Hernández-Orallo, 'A (hopefully) non-biased universal environment class for measuring intelligence of biological and artificial systems', in *Proceedings of AGI-10*, pp. 182–183, (2010).

[17] José Hernández-Orallo, 'AI Evaluation: past, present and future', *CoRR*, **abs/1408.6908**, (2014).

[18] José Hernández-Orallo and David L. Dowe, 'Measuring universal intelligence: Towards an anytime intelligence test', *Artificial Intelligence*, **174**(18), 1508–1539, (2010).

[19] Marcus Hutter. 50'000€ Prize for Compressing Human Knowledge, 2006.

[20] Benjamin Johnston, 'The toy box problem (and a preliminary solution)', in *Proceedings of AGI-10*, (2010).

[21] Yann LeCun and Corinna Cortes, *The MNIST database of handwritten digits*, 1998.

[22] Shane Legg and Marcus Hutter, 'Tests of Machine Intelligence', *CoRR*, **abs/0712.3825**, (2007). arXiv: 0712.3825.

[23] Shane Legg and Joel Veness, 'An approximation of the universal intelligence measure', in *Proceedings of the Ray Solomonoff 85th Memorial Conference*, volume 7070, pp. 236–249. Springer, (2011).

[24] John Levine, Clare Bates Congdon, Marc Ebner, Graham Kendall, Simon M. Lucas, Risto Miikkulainen, Tom Schaul, Tommy Thompson, Simon M. Lucas, and Michael Mateas, 'General Video Game Playing', *Artificial and Computational Intelligence in Games*, **6**, 77–83, (2013).

[25] *Beyond the Turing Test*, eds., Gary Marcus, Francesca Rossi, and Manuela Veloso, volume 37 of *AI Magazine*, AAAI, 1 edn., 2016.

[26] Pierre-Yves Oudeyer, Adrien Baranes, and Frédéric Kaplan, 'Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints', in *Intrinsically motivated learning in natural and artificial systems*, 303–365, Springer, (2013).

[27] Mark O. Riedl, 'The Lovelace 2.0 Test of Artificial Creativity and Intelligence', *CoRR*, **abs/1410.6142**, (2014).

[28] Robert Rosen, 'On models and modeling', *Applied Mathematics and Computation*, **56**(2), 359–372, (1993).

[29] Gavin A. Rummery and Mahesan Niranjan, 'On-line Q-learning using connectionist systems', Technical Report CUED/F-INFENG/TR 166, Cambridge University, (1994).

[30] Lorenza Saitta and Jean-Daniel Zucker, *Abstraction in Artificial Intelligence and Complex Systems*, Springer New York, New York, NY, 2013.

[31] Jürgen Schmidhuber, 'Formal theory of creativity, fun, and intrinsic motivation (1990–2010)', *Autonomous Mental Development, IEEE Transactions on*, **2**(3), 230–247, (2010).

[32] Daniel L. Scholten, 'Every good key must be a model of the lock it opens'. 2010.

[33] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others, 'Mastering the game of Go with deep neural networks and tree search', *Nature*, **529**(7587), 484–489, (2016).

[34] Satinder Singh, Andrew G. Barto, and Nuttapong Chentanez, 'Intrinsically Motivated Reinforcement Learning', in *Advances in Neural Information Processing Systems 17*, Vancouver, Canada, (2004).

[35] Kristinn R. Thórisson, 'Reductio ad Absurdum: On Oversimplification in Computer Science and its Pernicious Effect on Artificial Intelligence Research', in *Proceedings of AGI-13*, volume 7999 of *Lecture Notes in Computer Science*, Beijing, China, (2013). Springer.

[36] Kristinn R. Thórisson, Jordi Bieger, Stephan Schiffel, and Deon Garrett, 'Towards Flexible Task Environments for Comprehensive Evaluation of Artificial Intelligent Systems & Automatic Learners', in *Proceedings of AGI-15*, pp. 187–196, Berlin, (2015). Springer-Verlag.

[37] Kristinn R. Thórisson, Jordi Bieger, Thröstur Thorarensen, Jóna S. Sigurðardóttir, and Bas R. Steunebrink, 'Why Artificial Intelligence Needs a Task Theory — And What it Might Look Like'. arXiv: submit/1536181, 2016.

[38] Kristinn R. Thórisson and Eric Nivel, 'Achieving artificial general intelligence through peewee granularity', in *Proceedings of AGI-09*, pp. 220–221. Springer, (2009).

[39] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, and others, 'Stanley: The robot that won the DARPA Grand Challenge', *Journal of field Robotics*, **23**(9), 661–692, (2006).

[40] Alan M. Turing, 'Computing machinery and intelligence', *Mind*, **59**(236), 433–460, (1950).

[41] Pei Wang, 'The assumptions on knowledge and resources in models of rationality', *International Journal of Machine Consciousness*, **03**(01), 193–218, (2011).

[42] Shimon Whiteson, Brian Tanner, Adam White, and others, 'The reinforcement learning competitions', *AI Magazine*, **31**(2), 81–94, (2010).

[43] Michael Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2009.

:

Evaluation of General-Purpose Artificial Intelligence: Why, What & How

Jordi Bieger[1] **Kristinn R. Thórisson**1,[1,2] **Bas R. Steunebrink**[3]
Thröstur Thorarensen[1] and **Jóna S. Sigurðardóttir**[2]

Abstract. System evaluation allows an observer to obtain information about a system's behavior, and as such is a crucial aspect of any system research and design process. Evaluation in the field of artificial intelligence (AI) is mostly done by measuring a system's performance on a specialized task. This is appropriate for systems targeted at narrow tasks and domains, but not for evaluating general-purpose AI, which must be able to accomplish a wide range of tasks, including those not foreseen by the system's designers. Dealing with such novel situations requires general-purpose systems to be *adaptive*, learn and change over time, which evaluation based on quite different principles. The unique challenges this brings remain largely unaddressed to date, as most evaluation methods either focus on the binary assessment of whether some level of intelligence (e.g. human) has been reached, or performance on a test battery at a particular point in time. In this paper we describe a wide range of questions which we would like to see new evaluation methods for. We take look at various purposes for evaluation from the perspectives of different stakeholders (the *why*), consider the properties of adaptive systems that are to be measured (the *what*), and discuss some of the challenges for obtaining the desired information in practice (the *how*). While these questions largely still lack good answers, we nevertheless attempt to illustrate some issues that we believe are necessary (but perhaps not sufficient) to provide a strong foundation for evaluating general-purpose AI, and propose some ideas for directions in which such work could develop.

1 INTRODUCTION

Evaluation is the empirical means through which an observing system—an *evaluator*—obtains information about another system-under-test, by systematically observing its behavior. Evaluating general-purpose artificial intelligence (AI) is a challenge due to the combinatorial state explosion inherent in any system-environment interaction where both system and environment are complex. Furthermore, systems exhibiting some form of general intelligence must necessarily be highly adaptive and continuously learning (i.e. changing) in order to deal with new situations that may not have been foreseen during the system's design or implementation. Defining performance specifications for such systems is very different than doing so for systems whose behavior is not expected to change over time. Since the inception of the field of artificial intelligence (AI) the question of how to evaluate intelligence has puzzled researchers [27, 12, 10, 13, 23]. Many evaluation proposals to date have tried to transfer ideas from human testing [27, 7, 13], but this approach has severe limitations for artificial intelligence [2], where no single reference- or abstract system model can be assumed.

In this paper we discuss several important topics related to creating a solid foundation for evaluating (artificial) adaptive systems, organized around the three main topics of *why* we need special methods for evaluating adaptive systems, *what* should be measured when evaluating such systems, and *how* one might go about taking these measurements. For each one we highlight one or more topics that we see as critical yet unaddressed in the research literature so far. While having answers to many of the important questions raised in this paper would be desirable, we acknowledge that solutions remain out of reach to us, as much as our forerunners. For some we outline promising ways to address them, but for others we can only start by summarizing key issues and questions that must be answered in coming years (and decades).

Why might we want to evaluate adaptive systems? Numerous reasons could be cited, many of which will be shared by evaluation of other non-adaptive systems. Rather than try to be comprehensive in this respect we turn our attention here to three reasons for evaluating adaptive artificial systems that we feel are likely to lead to methods different from those developed for other kinds of systems: (a) testing whether performance levels in a particular range of areas are expected to be sufficient, (b) finding a system's strong and weak properties, and (c) establishing trust in a system by finding ways to predict its behavior. Different evaluators may wish to consider these aspects for various purposes: the system's designer may wish to find areas to improve, a teacher may wish to gauge training progress, a user may wish to deploy the system in situations where it will perform well, and potential adversaries may wish to exploit possible weaknesses. Understanding the relationship between task, environment, system, and evaluation methods is of critical importance as this will determine the appropriateness, efficiency, and meaningfulness with which any such measurements can be done.

What should be measured? Given that we are focusing on evaluation of general-purpose—and therefore adaptive—systems, it is somewhat surprising how research on this topic has tended to ignore its very central issue: *the adaptation process itself.* What adaptive systems have beyond other systems is that they change. Any proper test of adaptive systems must include a way to measure such *adaptivity*, including learning rate, knowledge retention, knowledge transfer, and sensitivity to interference, among other things. Yet most ideas on how to evaluate intelligent systems, starting with the Turing test and

[1] Center for Analysis and Design of Intelligent Agents,
School of Computer Science, Reykjavik University, Iceland.
email: {jordi13,thorisson,throstur11}@ru.is
[2] Icelandic Institute for Intelligent Machines, Reykjavik, Iceland.
email: jona@iiim.is
[3] The Swiss AI Lab IDSIA, USI & SUPSI, Manno-Lugano, Switzerland.
email: bas@idsia.ch

not changed much in character over the decades, has limited its scope to a measurement at a single point in time (see e.g. [13]).[4]

For the development of general—and beneficial—artificial intelligence, merely measuring current performance on a range of task-environments does not suffice. We must ascertain ourselves of the fact that our artificial adaptive system will be able to learn to deal with *novel* situations in a safe, beneficial and expedient way. Novelty calls for a kind of generality which to date remains to be successfully implemented in an artificial system. An ability that humans (and some animals) have and which seems central to general intelligence, and in particular important for novel environments and tasks, is *understanding*. Even within the field of artificial general intelligence (AGI) this special mechanism for adaptation seems to not have gotten the attention it deserves. It seems obvious that any proper evaluation method for intelligent systems must address understanding. In addition to performance under a variety of conditions, we must evaluate a system's robustness, learning/adaptation ability, and understanding of fundamental values. Unlike more specialized systems, where reliability in the specified range of situations suffices, we need to know that a general AI would adapt its behavior but not the core of its values in new situations.

So how can such things be measured? No consensus exists on what features adaptive systems should or must have, or what their purpose should be (nor can there be, since applications of such systems are countless): Looking for a single test, or even a standard battery of tests, for evaluating such systems is futile. Instead we argue that what is called for are a *task theory* [24] and a *test theory* [20], that would specify how construction of a *variety* of evaluation tests and methods can be done, as called for by the nature of the system to be evaluated and the aims of their developers.

In the remainder of this paper we will first discuss some background knowledge in section 2. Section 3, section 4 and section 5 will discuss the why, what and how of AI evaluation. Here we will consider the various purposes for which we might want to evaluate a system, identify various important fundamental and emergent properties of adaptive systems, and look at how we could obtain information about them. In section 6 we conclude the paper with a call for increased focus on the discussed areas of AI evaluation that have so far not received sufficient attention.

2 BACKGROUND

When we talk about intelligent systems under test, we can refer to either agents or controllers.[5] An agent consists of a (physical or virtual) body, containing its sensors and actuators, and a controller that acts as the "mind" of the system. In artificial intelligence research we are usually concerned with building ever more sophisticated controllers, while in robotics or applied AI we may also design the system's body. When we use the words "system", "actor" or "entity", we refer to whatever thing is being tested, whether that includes a body or not.

Intelligent systems interact with *task-environments*, which are tuples of a *task* and an *environment*. An *environment* contains objects that a system-under-test can interact with—which may form larger complex systems such as other intelligent agents—and rules that describes their behavior, interaction and affordances. A *state* is a configuration of these objects at some point or range in time. Tasks specify criteria for judging the desirability of states and whether or not they signify the successful or unsuccessful end of a task. The manner in which the task is communicated to the system-under-test is left open, and depends on the system and desired results of the evaluation. For instance, in (classical) AI planning the task is usually communicated to the system as a goal state at the start, while most reinforcement learners only get sporadic hints about what the task is through valuations of the current state.

The ultimate goal of evaluation is to obtain information about an intelligent system and its properties. This is done by observing its performance (behavior) as it interacts with a task-environment and/or the state that the task-environment is left in. For instance, we could evaluate a system just by the final score of a tennis match (of which evidence is left in the environment), or we could carefully analyze its behavior. Another example might be a multiple-choice exam, where we only look at the filled-out form at the end and don't consider the system's behavior over time. In a more elaborate written test, we may try to reproduce the system's thought process from the end result. Looking at final results is much easier, but also potentially much less informative as it throws out a lot of information.

Black-box evaluation methods look only at the input-output behavior of the system under test and its consequences, while white-box testing can also look at a system's internals. For fair and objective comparisons between different systems (e.g. humans and machines), black-box testing is typically desirable. Nevertheless, looking at gathered and utilized knowledge, or considering the performance of different modules separately can be quite informative—e.g.when debugging, finding weak points, or assessing understanding.

To define various properties of artificial systems to be measured, we must first have a decent understanding of the task-environments in which they are measured—preferably in the form of a *task theory* [24]. Task-environments—like intelligent systems—have both fundamental and emergent properties. For instance, the number of dimensions of a task-environment is an explicit (fundamental) part of its definition, whereas complexity emerges implicitly, and factors like observability and difficulty emerge in interaction with an intelligent system. We define the set of all task-environments to be $\mathbb{TE}$ and the set of properties to be $\mathbb{P}_{\mathbb{TE}} = \mathbb{L}_{\mathbb{TE}} \cup \mathbb{N}_{\mathbb{TE}}$, where $\mathbb{N}_{\mathbb{TE}}$ is for quantitative properties and $\mathbb{L}_{\mathbb{TE}}$ for qualitative ones[6]. A quantitative property $N \in \mathbb{N}_{\mathbb{TE}}$ is defined as a function from the set of all AI systems $\mathbb{A}$ and a collection of task environments to a real number: $N \in \mathbb{N}_{\mathbb{TE}} : \mathbb{A} \times \mathbb{TE}^n \rightarrow \mathbb{R}$. Collections of quantitative properties similarly map to a vector of real values: $\mathcal{N} \subset \mathbb{N}_{\mathbb{TE}} : \mathbb{A} \times \mathbb{TE}^n \rightarrow \mathbb{R}^n$. We define a distance metric $\mathcal{D} : \mathbb{TE} \times \mathbb{TE} \rightarrow [0, \infty)$, and $\mathcal{D}_{\mathcal{N} \subset \mathbb{N}_{\mathbb{TE}}}(X, Y) = f(\mathcal{N}(X), \mathcal{N}(Y))$, where $f : \mathbb{R}^n \times \mathbb{R}^n \rightarrow [0, \infty)$ can be any metric on $\mathbb{R}^n$; e.g. absolute/manhattan distance $f(\vec{x}, \vec{y}) = \sum_{i=0}^{|x|} |x_i - y_i|$

[4] Exceptions do of course exist, but given the importance of the subject, one would have expected the exact inverse ratio. See our earlier work on requirements for an evaluation framework for AI for a more in-depth discussion [23].

[5] As in control theory, we use the term "controller" to refer to control mechanisms in the broadest sense, irrespective of the methods they employ to achieve the control. An intelligent system's "controller" *includes anything that changes* during adaptation, such as memories, knowledge, know-how, reasoning processes, insight, foresight, etc., as well as the primary mechanisms instigating, managing and maintaining those changes. Any part of a system designated as belonging to its controller defines thus the *boundary* between *that which is being controlled* (e.g. a robot's body) and *that which does the controlling* (i.e. its "mind").

[6] Some examples of qualitative properties are the type of environment (e.g. is it a grid-based environment?), the nature of another agent (e.g. is it a friend, teacher, rival, etc.?), or the presence of particular phenomena (e.g. does it involve arithmetic?). In this paper we focus on quantitative aspects of evaluation however.

or Euclidean distance $\sqrt{\sum_{i=0}^{|x|}(x_i - y_i)^2}$. We similarly define the properties of adaptive systems $\mathbb{P}_{\mathbb{A}} = \mathbb{L}_{\mathbb{A}} \cup \mathbb{N}_{\mathbb{A}}$.

One defining aspect of AGI-aspiring systems is that they must *adapt* or *learn*: their knowledge and the behavior that follows from it change over time to better handle previously unknown situations. Here we take a very broad definition of knowledge that includes declarative knowledge (beliefs), procedural knowledge (skills), and structural knowledge (priorities). While the line between the core of a system and its (more fluid) knowledge can be blurry, it is occasionally useful to consider them separately. We define $\mathcal{K}(A)$ to be the knowledge of adaptive system $A \in \mathbb{A}$, and $\mathcal{K}(\mathcal{TE}, A, K_0, t_0 : t_n)$ to be the knowledge that system A with starting knowledge K_0 acquires/acquired in task-environments $\mathcal{TE} \subset \mathbb{TE}$ between times t_0 and t_n. An equivalent but alternative view is that K contains all of A's cognitive aspects that can change over time while A (also) contains its constants (e.g. its identity).

3 THE *WHY:* PURPOSES OF EVALUATION

Evaluation at its core is about obtaining information about the intelligent system-under-test. There are a number of reasons for why one might like to evaluate such a system. Evaluation can also be done by entities with a wide variety of relations to the system. Its developers may wish to improve its design, users may want to know what it can do (in which situations), teachers are interested in current knowledge levels and supported learning methods [4], and potential rivals may wish to size up the opposition. These parties have varying levels of control over the system under test and the evaluation process itself, and will need to take those limitations into account. Evaluations can be done in the lab, where the evaluators have full control over the task-environment and the system under test and can reset and tweak it at will, or they can be performed in the wild: e.g. by other agents who wish to interact with the system-under-test in some way.

3.1 Task-specific Performance

An often asked question—by consumers and creators alike—is whether a certain device is capable of performing a particular task that they need done, and if so, how well. Many AI systems are developed for a single, specific purpose and can often be evaluated in a relatively straightforward fashion. Performance is defined by the task at hand, and task-specific knowledge can be used to devise a model of the task-environment and/or select a collection of representative situations in which the system is to be tested.

Such evaluations are suitable in cases where the variation in the task-environment is well-known or can be controlled to a sufficient degree, and no real adaptation is required. This is typically not the situation in AGI research, where intelligent systems must be able to handle a wide range of tasks, both known and unknown at system design (and test) time. Nevertheless, even an AGI system may on occasion want to learn a particular task (in addition to the other tasks it already knows), in which case evaluation of task-specific performance could be appropriate for e.g. measuring training progress.

3.2 Strengths and Weaknesses

General cognitive abilities—and most generally *intelligence*—are used across a wide variety of tasks. Examples include the ability to reason by analogy, learn from examples, perform induction/abduction/deduction, respond in real-time, remember recent or long-ago events, understand causal chains, ignore distractions, etc. Knowledge of the levels of various cognitive abilities provides information about the system's strengths and weaknesses. This is useful for a variety of reasons:

- It points the system's developers to areas that need improvement.
- It can help users determine whether the system is suitable for (a range of) tasks or environments.
- It can help a teacher or friend find methods for education/interaction that the system will respond well to.
- It can help a potential adversary select strategies that avoid strengths and exploit weaknesses.

Depending on the evaluator's role, there are various amounts of control that they can exert over the system and the evaluation process.

In AI research we are mainly interested in *cognitive* capabilities, but generally speaking evaluation can also be used to test more physical capabilities.

3.3 Trust and Predictability

AGI systems are built to be deployed in situations that are not yet fully known when the system is designed and tested. Nevertheless, we would like to ensure that it behaves acceptably. To know this, we need to evaluate the range of situations in which it will behave according to specification. We can try to limit the system's exposure to situations that fit these parameters. When this is not possible or desirable, we want to know that the system degrades gracefully in difficult and/or novel situations and ideally that it will adapt to them and learn to perform well again over time. We also want to ensure that the system understands what "perform well" and "good outcome" mean, even in completely new situations.

For the development of general, beneficial artificial intelligence merely measuring performance on a range of task-environments does not suffice [20]. We must ascertain ourselves of the fact that it will be able to learn to deal with novel situations in a safe and beneficial way. To do so, we must evaluate robustness, learning/adaptation ability and understanding of fundamental values, as well as performance under various conditions. Unlike more specialized systems, where reliability in the specified range of situations suffices, we need to know that a general AI would adapt its *behavior* but not the core of its *values* in new situations [5]. To ensure good outcomes in an unpredictable, large, and complex (real) world, we need to look at a system's robustness, adaptivity and understanding.

4 THE *WHAT:* PROPERTIES OF INTELLIGENT SYSTEMS

After having identified different purposes of evaluation we can now turn to the various properties of artificial systems about which we may desire information. Some properties—such as the nature of the system's learning algorithms or its motivational system—are inherent in the system's design and may be amenable to inspection of its implementation, while other properties—such as the performance on a certain task or the amount of knowledge necessary to learn something—are emergent from the interaction with the world and are more amenable to evaluation. Although tests for qualitative aspects of a completely black-box system could in principle be designed, we focus here on the evaluation of quantitatively measurable properties.

4.1 Performance

For virtually all quantitative evaluations, some kind of *performance* measure is used as the main dependent variable. Too often however, all focus is placed on *precision*, *accuracy* or *correctness*, while measures of *efficiency* are relegated to secondary importance or ignored altogether. This can include the *speed* with which a task is performed (i.e. time efficiency), but also the reliance on other resources such as energy and knowledge (amount, diversity and content). The (independent) variable is often (training) time, if it is measured at all. These are important factors, and there are many others that can—and probably should—be considered as well.

We define *performance level* $\mathcal{P} \in \mathbb{N}_{\mathbb{A}} : \mathbb{A} \to [-1, 1]$ to be the main dependent variable for our evaluation purposes, where $\mathcal{P}$ can be some combination of accuracy/correctness $\mathcal{A}$, speed/time-efficiency $\mathcal{T}$, energy-efficiency $\mathcal{E}$, etc. The performance level of system A with knowledge K on task X can be written as $\mathcal{P}(X, A, K)$.[7] Efficiency/resource properties can also be defined with respect to a certain level of performance. For instance, $\mathcal{T}^{\mathcal{P}=0.9}$ could be the amount of training time required to reach a score of 0.9 on performance measure $\mathcal{P}$.

4.2 Adaptivity

To measure the adaptivity of a system, it is not only important to look at the rate at which a new task is learned, but also how much new knowledge is required.[8] The capacity for lifelong [26, 19] and transfer learning [22, 16, 11] depends not just on time, but on the content of old and new knowledge, as existing knowledge determines in part the speed of acquisition of new knowledge—both with respect to prerequisite knowledge already acquired (e.g. recognizing letters helps with learning to read) and to how related knowledge may apply to a new task, also called transfer learning (e.g. knowledge of driving one kind of motorized vehicle can help speed up learning how to drive others).

The most important measures of learning are probably the ones that relate the needed time, knowledge and other resources to a desired level of performance. Performance can for instance be measured as a function of training time, by varying t_n in $\mathcal{P}(X, A, \mathcal{K}(X, A, K_0, t_0 : t_n))$, which is the performance on task X after the system has trained on it between times t_0 and t_n ($\mathcal{K}$ is the knowledge system A with starting knowledge K_0 obtained in task-environment X between times t_0 and t_n). This can show how efficiently a certain task is learned. A more general measure of learning efficiency within a class of task-environments can be obtained by taking a weighted average of the performance on those other task-environments. However, this will only work if 1) the performance measures for various tasks are normalized (e.g. between -1 and 1), and 2) if corrections are made for the complexity and size of individual tasks. The goal here would be that if we encounter a new task with broadly the same properties as the measured class of task-environments, we can use the learner's general *learn rate* property to predict with some accuracy what would be needed to learn the new task (depending on known details such as its size and complexity).

[7] As before, we can think of K as the cognitive aspects of the actor A which can change over time, while A (also) contains constant aspects such as the system's identity.

[8] Knowledge acquisition takes time, so there is a correlation, but it is not perfect. Many algorithms spend much time processing the same data over and over, and the intelligence with which a space is explored can greatly influence how much new knowledge is gathered in a given time span (cf. active learning [18]).

Transfer learning ability $\mathcal{TL} : \mathbb{A} \times \mathbb{TE} \times \mathbb{TE} \to \mathbb{R}$ of a system from one task (collection) X to another Y can be defined in a number of complementary ways. For instance, we can look at how training for some time on X affects performance on Y in several ways. In each case we compare performance from training just on the target task(s) Y to performance of training on X first and then on Y.

- Raw performance transfer is the performance on Y after having trained on X for a given amount of time (dependent variable) or until a given level of performance: $\mathcal{P}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_n))$. This can also be considered as *generalization* to a different set of tasks. A special case would be if $Y \subset X$, in which case the test corresponds to a spot check of a larger amount of knowledge.
- The performance "loss" of training on the wrong task-environments can be calculated as the difference between $\mathcal{P}(X, A, \mathcal{K}(X, A, K_0, t_0 : t_n))$ and $\mathcal{P}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_n))$.
- A performance "gain" can be calculated by comparing how much "extra" training in another task-environment helps (or hinders): $\mathcal{P}(Y, A, \mathcal{K}(Y, A, \mathcal{K}(X, A, K_0, t_k : t_0), t_0 : t_n))$. Note that in this case more total time is used for training.
- Alternatively, we could look at whether it might help to spend part of a fixed time budget on task X before moving on to Y: $\mathcal{P}(Y, A, \mathcal{K}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_m), t_m : t_n))$.

Probably the most straightforward way to apply these measures is to take performance transfer as a function of training time t_n (and t_k/t_m). However, we can also take an extra step and analyze performance transfer as a function of attained performance level on X or the amount of knowledge that was acquired. Additionally, we can look at other things than performance, such as:

- Raw knowledge transfer is defined as the minimum amount of knowledge for reaching performance level y on Y that needs to be added to the system's knowledge after having trained on X: $|\mathcal{K}^{\mathcal{P}=y}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_n))|$.
- Training time transfer is defined as the difference between the amount of time to reach performance level y on Y from the current situation, and the amount of time needed to reach that level after having trained in X first: $\mathcal{T}^{\mathcal{P}=y}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_n))$.
- Composite time transfer is training time transfer plus the amount of time spent to train on X: $\mathcal{T}^{\mathcal{P}=y}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_n)) + t_n - t_0$.

Composite transfer measures can be used in teaching scenarios to judge whether it is worthwhile to decompose a task into component parts that are learned separately before a full task is presented [4].

In each case the transfer can be positive or negative. It is possible that knowledge is acquired in X that contradicts knowledge necessary to succeed in Y, possibly through no real fault of the system (e.g. it could have a bad teacher). Nevertheless, in many cases an intelligent adaptive system should be able to make use of its previously acquired knowledge when learning something new. It is therefore important that these systems retain some plasticity, even when they acquire more and more knowledge.

While it is most intuitive to consider transfer from previous tasks to newly learned ones, there can also be transfer (or interference) the other way around. Ideally, learning new tasks (e.g. a new language) should make one better at older tasks as well (e.g. other languages), but often the reverse is true. Catastrophic interference or forgetting plagues many machine learning systems: the ability to perform old tasks is lost when new tasks are learned [9, 3]. Interference and forgetting can be measured in similar methods as above.

So far we have assumed that one task (collection) is learned after another, but often tasks are learned and performed in parallel (cf. multitask learning [8, 17]). Again, similar measures can be defined for this scenario. In this case we also delve into the realm of distractions and robustness.

4.3 Robustness

Robustness is another important aspect of AI systems. The two main things to consider are *when* (or *if*) the system "breaks down" and *how* it does so. Furthermore, even in adverse or novel conditions we would like the system to eventually adapt so that it can properly function even in the new situation. Ideally we want a system that never breaks down, but this is likely not a realistic goal if we can not anticipate all the situations the system will find itself in. A more realistic goal may be to require that the system degrades gracefully, notices when things go awry and takes appropriate action—such as asking for help, moving back to a safer situation or gathering more information to start the adaptation process.

A general AI system may encounter various kinds of (internal and external) noise, distractions from extraneous input/output (dimensions) or parallel tasks, situations that differ on various dimensions from what it is used to, strain on its subsystems, or outright breakage of components. It is important to know that as these factors move further away from the ideal situation, the system will continue to function appropriately, detect the problem and/or degrade gracefully. Robustness can be measured with performance as the dependent variable and one or more kinds of interference as the independent variables. However, it can also be combined with other dependent variables such as training time or the ability to transfer knowledge between tasks. The standard form of measuring robustness is similar to knowledge transfer: $\mathcal{P}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_n))$. However, in this case we are more interested in the relation between task-environments X and Y than the training time and efficiency, and the difference between training on X first vs. training on Y directly. A good task-theory can help tremendously in the measurement of robustness. Most notably, we would want a task-environment generator or modifier that can create variants of the training environments X that differ in various desired ways.

One of the simplest notions of robustness is sensitivity to noise. Even a relatively primitive task-theory should make it possible to add noise, distortions or latency to the system's sensors and/or actuators. We could then draw a graph of how performance deteriorates as noise increases, which provides a nice quantitative picture of robustness in the face of a particular kind of interference. Other relatively easy-to-generate variations are to add irrelevant distractions to the environment (e.g. extra sensors, actuators or objects) or parallel tasks, to create a scarcity of resources (e.g. time, energy, knowledge).

If we look at $\mathcal{P}(Y, A, \mathcal{K}(X, A, K_0, t_0 : t_n))$ as a function of the distance between Y and X (measured along desired dimensions through some yet-to-be-invented task theory), we get a measure of *generalization*. Generalization ability of an adaptive system is among its most important properties, but we can additionally use these methods to judge the representativeness of certain training environments (X), which could then be used to more efficiently teach the system to perform well in a wide range of situations.

Aside from external sources of interference, we can also look at internal sources: what happens if faults occur inside of the system itself? If a (physical or "cognitive") component breaks down, will the system "die" or go "crazy", or will it just deteriorate performance slightly and perhaps even prompt the system to adjust and fix the problem? Some types of adversity that need not be catastrophic include memory deterioration or corruption, system/CPU strain/overload, synchronization errors, dropped messages, latency, noise, and failure of individual components (in a modular or distributed system).

In addition to looking at quantitative measures of dips in performance, it is also important to consider qualitative factors: if performance suddenly drops to zero we must ask what it means. Did the system just go crazy, or did it sensibly decide that the situation has deteriorated to the point where shutting down, warning a human or pursuing more fruitful endeavors is more appropriate? Answering such questions may require analyzing the system's behavior, motivations and/or reasoning in more detail.

4.4 Understanding

We typically want to see a certain continuity in our systems' behavior, even as they encounter new situations. For learning systems however, we also want them to adapt so that they may improve their performance even in these unforeseen situations. There is a delicate balance between change to a system's parameters that is desirable, and change that isn't. Importantly, we want performance to improve (or not degrade) from *our* perspective. Subjective improvement from the *system*'s perspective might be achieved by changing the way success is measured internally, but this is typically not something that we want. Most contemporary AI systems lack this capability, but more powerful and general systems of the future may possess the ability to recursively self-improve. While there are reasons to believe that a sufficiently intelligent system would attempt to protect the integrity of its goals [15, 6], we still need to ensure that these attempts are indeed successfully made.

Predictability results not just from vigorous quantitative tests, but also from more qualitative tests of a system's *understanding*. Some recent examples show that high performance on a task does not guarantee that the system performing it understands that task [21]. Deep neural networks have been trained to recognize images rather adequately—in some cases rivaling human-level performance—but are easily fooled with complete nonsense images or some slight deviations from the training data. When the stakes are higher, it is important to know that such weaknesses don't exist when situations differ slightly from the training scenario. By testing a system's understanding and examining its argumentation (cf. argument-based ML [14]), we can assure ourselves of the kind of reasoning that will be used even when novel situations are encountered. Perhaps even more importantly a system's ability to *grow* its understanding should be assessed to strengthen the foundation on which a system's level of adaptivity and intelligence is estimated, and the level of trust that we place in it.

Assessing a system's understanding of one or more phenomena[9] seems critical for generalizing a system's performance with respect to unfamiliar and novel tasks and environments. In prior work we have proposed a definition of understanding, based on the idea of models M of phenomena [25]. The closer the models describe important aspects of a phenomenon's properties and relations to other

[9] We define a phenomenon Φ (a process, state of affairs, thing, or occurrence) as $\Phi \subset W$ where W is the world in which the phenomenon exists and Φ is made up of a set of elements (discernible "sub-parts" of Φ $\{\varphi_1 \ldots \varphi_n \in \Phi\}$) of various kinds including relations $\Re_\Phi$ (causal, mereological, etc.) that couple elements of Φ with each other, and with those of other phenomena. See Thórisson et al. [25] for further details.

things the more general their utility, and the more deeply the system can be said to *understand* the phenomenon. Among other things, good models allow for making good predictions. Importantly, the theory goes further than this, however, requiring in addition that for proper assessment of a system's understanding its ability to explain, achieve goals with respect to, and (re-)create[10] a phenomenon must also be assessed.

In short, the theory states that, given any phenomenon Φ, model M_Φ contains information structures that together can be used to *explain* Φ, *predict* Φ, produce effective plans for *achieving goals* with respect to Φ, and *(re)create* Φ. For any set of models M, the closer the information structures $m_i \in M$ represent elements (sub-parts) $\varphi \in \Phi$, at any level of detail, including their internal and external relations/couplings $\Re_\Phi$, the greater the *accuracy* of M with respect to Φ. An adaptive system A's *understanding* of phenomenon Φ depends on the quality, that is *accuracy* and *completeness* of A's *models M* of Φ, which enable prediction, action upon, explanation, and (re)creation of Φ. The better such models describe Φ, the better any of these will be. Understanding thus has a (multidimensional) *gradient* from low to high levels [25].

Prediction is one form of evidence for understanding. Some prediction can be done based on correlations, as prediction does not require representation of the direction of causation yet captures co-occurrence of events. Prediction of a particular turn of events requires (a) setting up initial variables correctly, and (b) simulating the implications of (computing deductions from) this initial setup.

A number of different questions can be asked regarding the prediction of a phenomenon, for instance:

- From a particular (partial) start state, what is the time (range) in which the phenomenon is expected to occur (if at all)?
- What will be the state of the phenomenon at a future time, given some starting conditions?
- For some phenomenon $\Phi = \{\phi_1 \ldots \phi_n\}$, given values for some subset $\Psi \subset \Phi$, predict the values for the remaining $\phi_i \in \Phi$.
- Predict the state or occurrence of related phenomena $\Omega \subset \Re_\Phi$ given the state of Φ.

Picking an appropriate set of such questions is at the heart of properly evaluating a system's ability to predict a phenomenon.

Goal Achievement. Correlation is not sufficient, however, to inform how one achieves goals with respect to some phenomenon Φ. For this one needs causal relations. Achieving goals means that some variables in Φ can be manipulated directly or indirectly (via intermediate variables). Achieving goals with respect to a phenomenon Φ does not just require understanding the individual components of Φ itself, but also how these relate to variables that are *under the system's control*. In short: the system needs models for interaction with the environment as well as the phenomenon. For a robotic agent driving a regular automobile, to take one example, the system must possess models of its own sensors and manipulators and how these relate to the automobile's controls (steering wheel, brakes, accelerator, etc.). Such interfaces tend to be rather task-specific, however, and are thus undesirable as a required part of an evaluation scheme for understanding. Instead, we call for an ability to *produce effective plans* for achieving goals with respect to Φ. An effective plan is one that can be proven useful, efficient, effective, and correct, through implementation.[11]

Goal achievement with respect to some phenomenon Φ can be defined by looking at the system's performance (cf. section 4.1) in task-environments and/or interactions that feature Φ. The phenomenon can play a number of different roles, depending on its type (e.g. event, process, tool, obstacle, etc.). Events can be caused, prevented or changed (usually within a certain time range). Objects can have their state configured in a desired manner. When an object is a tool or obstacle, we can compare the performance in environments with and without Φ. Processes can have several effects on the environment (possibly depending on the manner of their execution), and we can set a task-environment's goal to be accomplished by some of these effects and negated by others to see if the system can flexibly execute the process. If the system's performance in task-environments and/or interactions that include Φ are consistently better than when Φ is absent, this can indicate a higher level of understanding of Φ.

Explanation is an even stronger requirement for demonstrating understanding, testing a system's ability to use its models for abductive reasoning. Correlation does not suffice for producing a (true) explanation for an event or a phenomenon's behavior, as correlation does not imply causation. One may even have a predictive model of a phenomenon that nevertheless represent incorrectly its parts and their relations (to each other and parts of other phenomena). This is why scientific models and theories must be both predictive *and* explanatory—together constituting a litmus test for complete and accurate capturing of causal relations.

(Re)creating a phenomenon is perhaps the strongest kind of evidence for understanding. It is also a prerequisite for correctly building new knowledge that relies on it, which in turn is the key to growing one's understanding of the world. By "creating" we mean, as in the case of noted physicist Richard Feynman, the ability to produce a model of the phenomenon in sufficient detail to replicate its necessary and sufficient features. Note that this is not limited to (re)creation *by the system* using its own I/O capabilities, but involves an understanding of how the phenomenon can be created *in general* by the system, by others, by the environment itself, or even by some hypothetical entity with (partially) imagined capabilities. Requiring understanders to produce such models exposes the completeness of their understanding.

It is important to emphasize here that understanding, in this formulation, is not reductionist: Neither does it equate the ability to understand with the ability to behave in certain ways toward a phenomenon (e.g. achieve goals), nor the ability to predict it, nor the ability to explain it, nor the ability to (re)create it. While any of these may provides hints of a system's understanding of a phenomenon, it cannot guarantee it. In our theory *all are really required* (to some minimum extent) to (properly) assess a system's understanding; any assessment method that does not include these four in some form runs a significantly higher risk of failure.

5 THE *HOW:* CONDUCTING TESTS AND ANALYZING RESPONSES

All evaluations are contextual: i.e. they are done with respect to a task-environment, or collection of task-environments. We should examine how the measurements depend on the chosen collection of task-environments, and strive towards using as large a range as possible. We will need to say something about the range of task-environments that we think our results generalize to as well. Se-

[10] We mean this in the same sense as when we say that a chef's recipe demonstrates her understanding of baking, or a physicists' simulation of the universe demonstrates their understanding of how the universe works.

[11] Producing plans, while not being as specific as requiring intimate familiarity with some I/O devices to every Φ, requires nevertheless knowledge of some language for producing said plans, but it is somewhat more general and thus probably a better choice.

lection and/or creation of task-environments for the optimal measurement of desired system properties that generalize to other task-environments requires a *task theory* [24].

5.1 Construction and Selection of Task-Environments

We need a way to relate the things we want to test for to the information that can be obtained from (aspects of) task-environments. Due to the differences in AI systems, the purposes for which they are built and the properties we want to test for, it is impossible to construct a single test, or test battery, that measures everything for all systems. Rather, we need a task theory that allows us to analyze and construct tasks to specification, in combination with knowledge about the properties and behavior of intelligent, adaptive systems.

Given an intelligent system and a question about one or more of its properties, we should be able to a) construct a task-environment, b) adapt a given task-environment, or c) select a task-environment from some choices, in order to optimally answer that question about the system. Given a task-environment we would like to be able to predict reasonable behaviors for a certain system or class of systems with certain properties. An informative task-environment would afford multiple behaviors that are distinctive with respect to the property that we want to measure. It is likely most informative to test around the edges/limits of the system's capabilities. A task theory that allows for scaling the scope or difficulty of environments would therefore be tremendously useful [23].

In some cases it may be possible to construct batteries of tasks for answering a certain question about a set of systems—e.g. a standardized exam. In other cases the evaluation may be more interactive and explorative. Another important consideration is how much control we have over the system (e.g. can we look at its source code or memory?) and its task-environment (e.g. is it virtual and owned by us?).

Motivation and Incentive Somehow we need to get the system to actually perform the envisioned task, which may be difficult without full control. Simply placing a system in a task-environment doesn't guarantee that it will perform (or even understand) the task that we want. If you place a child in a room with a multiple-choice IQ test, will it fill it out as you want? Or will it check the boxes in an aesthetically pleasing manner? Or just ignore the test? In general we can never be sure, but we can try to incentivize good performance on the test. Alternatively, we can look at the behavior and try to derive the task the system was trying to perform (cf. inverse reinforcement learning [1]; although this tends to assume a certain level of competency on the part of the system).

5.2 Judging Behavior

Rather than just looking at end results (e.g. score on an exam or tennis match), we can also look at performance/behavior during the test (i.e. the sequence of actions in response to stimuli). This should hopefully shed some light on inner workings and allow us to construct a model that is predictive in more situations.

In situations where we know a good solution to a task, we can compare that solution (or those solutions) to the observed behavior of the system. Assuming the system has the appropriate goals, we can then see where it deviates and consider what gap in knowledge or leap in reasoning led it to do so. Alternatively, under the assumption that the system is reasonably competent, we can try to find its motivations and goals through inverse reinforcement learning [1].

Deconstruction/decomposition of tasks into multiple smaller parts can be extremely useful for this purpose. In that case, we can use easier-to-perform performance evaluations on a much more granular scale.

5.3 Evaluating Understanding

To test for evidence of understanding a phenomenon Φ (a process, state of affairs, thing, or occurrence) in a particular task environment, we may probe (at least) four capabilities of the system (a) to *predict* Φ, (b) to *achieve goals* with respect to Φ, (c) to *explain* Φ, and (d) to *(re)create* Φ [25]. All can have a value in $[0, 1]$ where 0 is no understanding and 1 is perfect understanding.[12] For a thorough evaluation of understanding all four should be applied.

The major challenge that remains is how to perform this assessment. Goal achievement can be measured in a reasonably straightforward fashion, although we do require a way to construct goals and tasks that incorporate the phenomenon for which understanding is to be tested. Similarly, it should be possible to define a task that involves the desired phenomena's recreation. Testing for high-level predictions seems more challenging if the system doesn't automatically communicate the predictions that it makes. Somewhat imperfect tests for predictions can be constructed by presenting the system with situations where correct predictions would likely prompt it to show different behavior than incorrect predictions. Alternatively, it may be possible to access the system's internals, in which case a trace of its operation may show which events and observations were expected.

Measuring explanations may be the most important and difficult challenge in AI evaluation though. Most systems are not explicitly built to provide human-understandable explanations for their actions, but from this we cannot conclude that they are not adequately modeling the causal chains of the environment and justifying their behavior to themselves in some way. If a system doesn't explicitly try to explain itself, then it seems that we can only access explanations by inspecting the system's inner workings. Subsymbolic systems are notoriously difficult to understand for humans, but even symbolic systems could present difficulties; either because their symbols are unlabeled and grounded in a different ways than ours, or because the amount of involved models and considerations in each decision are overwhelming. Overcoming these issues is an open problem, but given the importance of modeling the world's causal chains and making justifiable decisions, we suggest that AGI systems ought to be built with a faculty for explanation and summarization in mind, which should help us evaluate their understanding.

6 CONCLUSION & FUTURE WORK

Evaluation of intelligent adaptive systems is important for a wide variety of reasons. Progress in AI depends on our ability to evaluate it: to find the strengths and weaknesses of our programs and improve them where necessary. Looking at performance alone is not enough, since we need our more general systems to operate beneficially even

[12] More complex measurements could of course be used for a more thorough or faithful representation of understanding; projecting it down to a single dimension may lose some (important) information. This simplification is however immaterial to the present purposes.

in situations that we did not fully foresee. We must therefore consider these systems' robustness to changing and possibly deteriorating conditions and acquire confidence that they will adapt in ways that allow them to continue to be beneficial to their human owners.

Focus in AI evaluation has been mostly on testing for performance—often in specialized (and limited) domains—by measuring some final result that was attained on a task at a single point in time. Not only do we need to consider other factors like adaptivity and robustness: we must also look beyond the final impact that is made on the system's environment. Moment-to-moment behavior can be a rich source of information that sheds much light on how or why a certain level of performance was attained. Even more importantly, we must attempt to measure levels of understanding. An explanation is more than a single data point: it is a model that can be applied in many situations. If we know that a system understands certain concepts—most notably our values—we can be relatively confident that it will make the right considerations, even in unforeseen situations.

Measuring system properties beyond performance as well as the analysis of behavior and understanding are very challenging, and it is not obvious how to do it. It is however clear that better theories for testing, understanding and task-environments are a part of the solution. Future work must investigate these avenues of research that are necessary if we are to move forward in our quest for general-purpose adaptive AI.

ACKNOWLEDGEMENTS

The authors would like to thank Magnús Pálsson at IIIM for comments on an earlier version of this paper. This work was sponsored by the School of Computer Science at Reykjavik University, and a Centers of Excellence Grant (IIIM) from the Science & Technology Policy Council of Iceland.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng, 'Apprenticeship learning via inverse reinforcement learning', in *Proceedings of the twenty-first international conference on Machine learning*, p. 1. ACM, (2004).
[2] Tarek Besold, José Hernández-Orallo, and Ute Schmid, 'Can Machine Intelligence be Measured in the Same Way as Human intelligence?', *KI - Künstliche Intelligenz*, 1–7, (April 2015).
[3] J. Bieger, I. G. Sprinkhuizen-Kuyper, and I. J. E. I. van Rooij, 'Meaningful Representations Prevent Catastrophic Interference', in *Proceedings of the 21st Benelux Conference on Artificial Intelligence*, pp. 19–26, Eindhoven, The Netherlands, (2009).
[4] Jordi Bieger, Kristinn R. Thórisson, and Deon Garrett, 'Raising AI: Tutoring Matters', in *Proceedings of AGI-14*, pp. 1–10, Quebec City, Canada, (2014). Springer.
[5] Jordi Bieger, Kristinn R. Thórisson, and Pei Wang, 'Safe Baby AGI', in *Proceedings of AGI-15*, pp. 46–49, Berlin, (2015). Springer-Verlag.
[6] Nick Bostrom, 'The superintelligent will: Motivation and instrumental rationality in advanced artificial agents', *Minds and Machines*, **22**(2), 71–85, (2012).
[7] Selmer Bringsjord and Bettina Schimanski, 'What is artificial intelligence? Psychometric AI as an answer', in *IJCAI*, pp. 887–893. Citeseer, (2003).
[8] Richard A. Caruana, *Multitask Learning Thesis*, PhD, Carnegie Mellon University, Pittsburgh, PA, September 1997.
[9] Robert M. French, 'Catastrophic interference in connectionist networks', in *Encyclopedia of Cognitive Science*, ed., Lynn Nadel, volume 1, 431–435, Nature Publishing Group, London, (2003).
[10] José Hernández-Orallo, 'AI Evaluation: past, present and future', *CoRR*, **abs/1408.6908**, (2014).
[11] Alessandro Lazaric, 'Transfer in Reinforcement Learning: A Framework and a Survey', in *Reinforcement Learning*, 143–173, Springer, (2012).
[12] Shane Legg and Marcus Hutter, 'Tests of Machine Intelligence', *CoRR*, **abs/0712.3825**, (2007). arXiv: 0712.3825.
[13] *Beyond the Turing Test*, eds., Gary Marcus, Francesca Rossi, and Manuela Veloso, volume 37 of *AI Magazine*, AAAI, 1 edn., 2016.
[14] Martin Možina, Jure Žabkar, and Ivan Bratko, 'Argument based machine learning', *Artificial Intelligence*, **171**, 922–937, (2007).
[15] Stephen M. Omohundro, 'The basic AI drives', *Frontiers in Artificial Intelligence and applications*, **171**, 483, (2008).
[16] Sinno Jialin Pan and Qiang Yang, 'A Survey on Transfer Learning', *IEEE Transactions on Knowledge and Data Engineering*, **22**(10), 1345–1359, (October 2010).
[17] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov, 'Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning', *arXiv:1511.06342 [cs]*, (November 2015). arXiv: 1511.06342.
[18] Burr Settles, 'Active learning', *Synthesis Lectures on Artificial Intelligence and Machine Learning*, **6**(1), 1–114, (2012).
[19] Daniel L. Silver, Qiang Yang, and Lianghao Li, 'Lifelong Machine Learning Systems: Beyond Learning Algorithms.', in *AAAI Spring Symposium: Lifelong Machine Learning*, (2013).
[20] Bas R. Steunebrink, Kristinn R. Thórisson, and Jürgen Schmidhuber, 'Growing recursive self-improvers', *To be published in B. R. Steunebrink et al. (eds.), Proceedings of Artificial General Intelligence 2016*, (2016).
[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, 'Intriguing properties of neural networks', *arXiv:1312.6199 [cs]*, (December 2013). arXiv: 1312.6199.
[22] Matthew E. Taylor and Peter Stone, 'Transfer learning for reinforcement learning domains: A survey', *The Journal of Machine Learning Research*, **10**, 1633–1685, (2009).
[23] Kristinn R. Thórisson, Jordi Bieger, Stephan Schiffel, and Deon Garrett, 'Towards Flexible Task Environments for Comprehensive Evaluation of Artificial Intelligent Systems & Automatic Learners', in *Proceedings of AGI-15*, pp. 187–196, Berlin, (2015). Springer-Verlag.
[24] Kristinn R. Thórisson, Jordi Bieger, Thröstur Thorarensen, Jóna S. Sigurðardóttir, and Bas R. Steunebrink, 'Why Artificial Intelligence Needs a Task Theory — And What it Might Look Like', *To be published in B. R. Steunebrink et al. (eds.), Proceedings of Artificial General Intelligence 2016*, (2016). arXiv: 1604.04660.
[25] Kristinn R. Thórisson, David Kremelberg, and Bas R. Steunebrink, 'About understanding', *To be published in B. R. Steunebrink et al. (eds.), Proceedings of Artificial General Intelligence 2016*, **6**, (2016).
[26] Sebastian Thrun, 'Lifelong Learning: A Case Study', Technical CMU-CS-95-208, Carnegie Mellon University, Pittsburgh, PA, (November 1995).
[27] Alan M. Turing, 'Computing machinery and intelligence', *Mind*, **59**(236), 433–460, (1950).

DisCSP-Netlogo- an open-source framework in NetLogo for implementation and evaluation of the distributed constraints

Ionel Muscalagiu[1], Popa Horia Emil[2] and Jose Vidal [3]

Abstract. Distributed Constraint programming (DisCSP/DCOP) is a programming approach used to describe and solve large classes of problems such as searching, combinatorial and planning problems. A simulation framework in NetLogo for distributed constraints search and optimization algorithms is presented. The purpose of this paper is to present an open-source solution for the implementation and evaluation of the distributed constraints in NetLogo. This tool can run with or without a graphical user interface in a cluster of computers with a large number of agents. It includes all needed techniques for implementing all existing DCSP and DCOP algorithms. A comparison with the main evaluation and testing platforms for distributed constraints search and optimization algorithms is presented.

1 Introduction

Constraint programming is a programming approach used to describe and solve large classes of problems such as searching, combinatorial and planning problems. A Distributed Constraint Satisfaction Problem (DisCSP) is a constraint satisfaction problem in which variables and constraints are distributed among multiple agents [16], [7]. A Distributed Constraint Optimization Problem (DCOP) is similar to the constraint satisfaction problem except that the constraints return a real number instead of a Boolean value and the goal is to minimize the value of these constraint violations.

Distributed Constraint Satisfaction/Distributed Constraint Optimization is a framework for describing a problem in terms of constraints that are known and enforced by distinct participants (agents). This type of distributed modeling appeared naturally for many problems for which the information was distributed to many agents. DisCSPs are composed of agents, each owning its local constraint network. Variables in different agents are connected by constraints forming a network of constraints. Agents must assign values to their variables so that all constraints between agents are satisfied. Instead, for DCOP a group of agents must distributedly choose values for a set of variables so that the cost of a set of constraints over the variables is either minimized or maximized. Distributed networks of constraints have proven their success in modeling real problems.

There are some algorithms for performing distributed search in cooperative multiagent systems where each agent has some local information and where the goal is to get all the agents to set themselves to a state such that the set of states in the system is optimal. There exist complete asynchronous searching techniques for solving the DisCSP in this constraints network, such as the ABT (Asynchronous Backtracking), AWCS (Asynchronous Weak Commitment) [16], ABTDO (Dynamic Ordering for Asynchronous Backtracking) [7], AAS (Asynchronous Search with Aggregations) [14], DisDB (Distributed Dynamic Backtracking) [2] and DBS (Distributed Backtracking with Sessions) [9].

Also, for DCOP there are many algorithms among whom we name ADOPT (Asynchronous Distributed OPTimization) [8] or DPOP (Dynamic Programming Optimization Protocol) [12]. We find that many multiagent problems can be reduced to a distributed constraints problem. Many problems in the areas of computer science, engineering, biology can be modeled as constraint satisfaction problems (or distributed CSP). Some examples include: spatial and temporal planning, diagnosis, decision support, hardware design and verification, real-time systems and robot planning, protein structure prediction problem, DNA structure analysis, timetabling for hospitals, industry scheduling, transport problems, etc.

The implementation and testing of the asynchronous search techniques implies a programming effort not at all trivial. Thus, the necessity of developing a dedicated platform that can be used for testing them became a necessity. There are some platforms for implementing and solving DisCSP problems: DisChoco [19], DCOPolis [13], DisCo [4], FRODO [5] and DisCSP-Netlogo [10, 21]. Such a tool allows the use of various search techniques so that we can decide which is the most suitable one for that particular problem. Also, these tools can be used for the study of agents' behavior in several situations, like the priority order of the agents, the synchronous and asynchronous case, apparition of delays in message transmission, therefore leading to identifying possible enhancements of the performances of asynchronous search techniques.

The asynchronous search techniques involves concurrent (distributed) programming. The agents can be processes residing on a single computer or on several computers, distributed within a network. The implementation of asynchronous search techniques can be done in any programming language allowing a distributed programming, such as Java, C/MPI or other. Nevertheless, for the study of such techniques, for their analysis and evaluation, it is easier and more efficient to implement the techniques under a certain distributed environment, such as the new generation of multiagent modeling language (NetLogo [17], [21], [22], [6]).

NetLogo is regarded as one of the most complete and successful agent simulation platforms [17], [6]. NetLogo is a high-level platform, providing a simple yet powerful programming language, built-in graphical interfaces and the necessary experiment visualization

[1] Politehnica University of Timisoara, Romania, email: mionel@.fih.upt.ro.
[2] The University of the West, Timisoara, Romania, email:hpopa@info.uvt.ro
[3] Computer Science and Engineering, University of South Carolina, USA, email:vidal@sc.edu

tools for quick development of simulation user interface. It is a environment written entirely in Java, therefore it can be installed and activated on most of the important platforms.

The purpose of this paper is to present an open-source solution for implementation and evaluation of the asynchronous search techniques in NetLogo, for a great number of agents, model that can be run on a cluster of computers. However, this model can be used in the study of agents' behavior in several situations, like the priority order of the agents, the synchronous and asynchronous case, etc. Our goal is to supply the programmer with sources that can be updated and developed so that each can take profit from the experience of those before them. In fact, it is the idea of development adopted in the Linux operating system. Any researcher has access to the existing implementations and can start from these for developing new ones. The platform offers modules for various problems of evaluation such as: the random binary problems, random graph coloring, multi-robot exploration.

This paper synthesizes all the tries of modeling and implementation in NetLogo for the asynchronous search techniques [10],[11]. Many implementations were done for a class of algorithms from the ABT and AWCS families (DisCSP), respectively ADOPT (DCOP). They can be downloaded from the websites [21], [22].

The DisCSP-NetLogo modules were designed for the implementation, learning and evaluation of distributed algorithms [10, 11]. These modules offer support for researchers from the DisCSP/DCOP domain for developing their algorithms, for evaluating the performances of these algorithms, and why not, for tutorials that allow teaching these techniques that have a pretty high degree of difficulty. Also, these modules allow the study of the agents' behavior, visualization of various values attached to the agents and, as a consequence, allowing us to understand these complex systems.

The DisCSP Netlogo allows various extensions: the modules can be updated and extended. The test problem generators can be particularized and new problem types can be added. Extreme situations can be simulated, such as network delays, allowing users to test the same algorithms under different network conditions. The DisCSP Netlogo allows the running with GUI also the visualization in real time of the metrics associated with the algorithm. The visualization of partial solutions is also possible with this tool. By visualization of various aspects of distributed search algorithms, one can attain new insights on the behavior of these algorithms. These insights may help drive new variants of search algorithms and different heuristics for these algorithms. The visualization also enables algorithm debugging during the implementation process of new algorithms, and can serve as an educational method which dynamically displays an algorithms progress.

We adapted the HubNet technology to allow the agents to run on computers or mobile devices from the local network or over the Internet. A template model is provided, this template can be downloaded from the websites [21]. This thing will allow us to run the agents in conditions as close as possible to the real ones, opposite to the majority of dedicated platforms that allow an evaluation in the simulated mode. Typically, the DisCSP/DCOP algorithms have been evaluated along two dimensions: computation and communication. The solution proposed here will allow to recreate more realistic scenarios and to better understand algorithms behavior in conditions of existing network latency. This tool is aimed to allow the evaluation of distributed algorithms in conditions as similar as possible to the real situations.

Some problems modeled with distributed constraints are exemplified:

- the randomly generated problem that has a structure of scale-free network (the constraint graph has a structure of scale-free network) .
- the multi-robot exploration problem.
- the randomly generated (binary) CSPs.
- the protein folding problem.

2 Modeling and implementing of the asynchronous search techniques in NetLogo

In this section we present a solution of modeling and implementation for the existing agents' process of execution in the case of the asynchronous search techniques. This open-source solution, called DisCSP-NetLogo is extended so that it is able to run on a larger number of agents, model runnable on a cluster of computers and is presented below. This modeling can also be used for any of the asynchronous search techniques, such as those from the AWCS family [16], ABT family [2], DisDB [2], DBS [9]. Implementation examples for these techniques can be found on the sites in [21],[22].

The modeling of the agents' execution process is structured on two levels, corresponding to the two stages of implementation [10], [20]. The definition of the way in which asynchronous techniques are programmed so that the agents run concurrently and asynchronously constitutes the internal level of the model. The second level refers to the way of representing the surface of the implemented applications. This is the exterior level.

In any NetLogo agent simulation, four entities (objects)participate:

- *The Observer*, that is responsible for simulation initialisation and control. This is a central agent.
- *Patches*, i.e. components of a user defined static grid (world) that is a 2D or 3D world, which is inhabited by turtles. Patches are useful in describing environment behavior.
- *Turtles* that are agents that "live" and interact in the world formed by patches. Turtles are organised in breeds, that are user defined groups sharing some characteristics, such as shape, but most importantly breed specific user defined variables that hold the agents' state.
- *Links* agents that "connect" two turtles representing usually a spatial/logical relation between them.

2.1 Agents' simulation and initialization

First of all, the agents are represented by the breed type objects (those are of the turtles type). In Figure 1 is presented the way the agents are defined together with the global data structures proprietary to the agents. We implement in open-source NetLogo the agents' process of execution in the case of the asynchronous search techniques [10],[20]:

S1. Agents' simulation and initialization in DisCSP-NetLogo. First of all, the agents are represented by the breed type objects (those are of the turtles type). Figure 1 shows the way the agents are defined together with the global data structures proprietary to the agents.

This type of simulation can be applied for different problems used at evaluation and testing:

-the distributed problem of the n queens characterized by the number of queens.

-the distributed problem of coloring of a randomly generated graph characterized by the number of nodes, colors and the number of connections between the nodes.

```
breeds [agents]
globals[variables that simulate the memory shared by all the agents]
agent-own [Message-queue Current-view MyValue Nogoods
nr-constraintc messages-received-ok ... AgentC-Cost]
;Message-queue contains the received messages.
;Current-view is a list indexed on the agent's number, of the form [v0 v1...],
;vi = -1 if we don't know the value of that agent.
;nogoods is the list of inconsistent value [0 1 1 ... ], where 1 is inconsistent.
;messages-received-ok,count the number of messages received by an agent.
;nr-cycles -the number of cycles,
;AgentC-Cost - a number of non-concurrent constraint checks
```

Figure 1. Agents' definition in DisCSP-Netlogo for the asynchronous search techniques

- **-the randomly generated (binary) CSPs** characterized by the 4-tuple (n, m,p1,p2), where: n is the number of variables; m is the uniform domain size; $p1$ is the portion of the $n \cdot (n-1)/2$ possible constraints in the constraint graph; $p2$ is the portion of the $m \cdot m$ value pairs in each constraint that are disallowed by the constraint.
- **-the randomly generated problem that has a structure of scale-free network** (the constraint graph has a structure of scale-free network) [1]. An instance DisCSP that has a structure of scale-free network have a number of variables with a fixed domain and are characterized by the 5-tuple (n, m, t,md, γ), where *n* is the number of variables, *m* is the domain size of each variable; *t* (the constraint tightness) determining the proportion of value combinations forbidden by each constraint, md=the minimal degree of each nodes and γ is the exponent that depends on each network structure. A scale-free network is characterized by a power-law degree distribution as follows $p(k) \propto k^{-\gamma}$ [1].
- **-the multi-robot exploration problem [3]** are characterized by the 6-tuple (n, m, p1, sr, cr, obsd), where:
  - n is the number of robots exploring an environment, interact and communicate with their spatial neighbors and share a few common information (information about already explored areas);
  - $m = 8$ is the domain size of each variable; Dom(x_i)is the set of all 8 cardinal directions that a robot A_i can choose to plan its next movement.
  - $p1$ - network-connectivity, sr - the sensor range of a robot, cr - the communication range of a robot;
  - $obsd$ - obstacles-density. We have considered environments with different levels of complexity depending on: the number of obstacles, the size of the obstacles, the density of the obstacles.

For these types of problems used in the evaluation there are NetLogo modules that can be included in the future implementations. The modules are available on the website [21]. For each module are available procedures for random generation of instances for the choosen problems, together with many ways of static ordering of the agents. Also, there are procedures for saving in files the generated instances and reusing them for the implementation of other asynchronous search techniques. On the website [21] can be found many modules that can generate problem instances (both solvable and unsolvable problems) with various structures for the previous problems, depending on various parameters (uniform random binary DisCSP generator, scale-free network instance generator for DisCSP). An example is presented in Figure 2 for the random binary problems.

S2.Representation and manipulation of the messages. Any asynchronous search technique is based on the use by the agents of some messages for communicating various information needed for obtaining the solution. The agents' communication is done according to the communication model introduced in [16].

```
breeds [agents-nodes]
breeds [edges]
;nodes = agents, each undirected edge goes from a to b
;edges = links agents that connect two agents-nodes
;representing usually a spatial/logical relation between them.
globals[Orders done nr-cycles domain-colour-list no-more-messages]
agent-own [Message-queue Current-view MyValue Nogoods ChildrenA
ParentA nr-constraintc messages-received-ok ... AgentC-Cost]

__includes["RBP.nls" "StaticOrders.nls"]
;are included the modules for generating instances and choosing
a static order for the agents ...
to setup ; Setup the model for a run, build a constraints graph.
  setup-globals ; setup Global Variables
  setup-patches ; initialize the work surface on which the agents move
  setup-turtles we generate the objects of the turtles type that simulate the agents
  setup-random-binary-problems ; or LoadRBRFile
  ; we generate a the types of problems used at the evaluation
  ; or we load an instance generated and salved previously.
  CalculateOrdersMaxCardinality ; is selected from variable-ordering heuristics
  setup-DisCSP
;we initialize the data structures necessary for the DisCSP algorithm
end
```

Figure 2. Templates for agents' definition in DisCSP-Netlogo for the random binary problems

The communication model existing in the DisCSP frame supposes first of all the existence of some channels for communication, of the FIFO type, that can store the messages received by each agent. The way of representation of the main messages is presented as follows:

◦ (list "type message" *contents* Agent-costs) ;

The simulation of the message queues for each agent can be done using Netlogo lists, for whom we define treatment routines corresponding to the FIFO principles. These data structures are defined in the same time with the definition of the agents. In the proposed implementations from this paper, that structure will be called message-queue. This structure property of each agent will contain all the messages received by that agent.

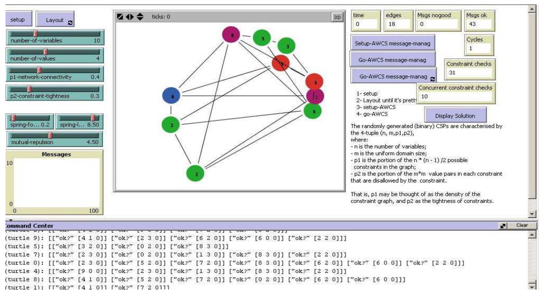
The manipulation of these channels can be managed by a central agent (which in NetLogo is called observer) or by the agents themselves. In this purpose we propose the building of a procedure called *go* for global manipulation of the message channels. It will also have a role in detecting the termination of the asynchronous search techniques' execution. That *go* procedure is some kind of a "main program", a command center for agents. The procedure should also allow the management of the messages that are transmitted by the agents. It needs to call for each agent another procedure which will treat each message according to its type. This procedure will be called handle-message, and will be used to handle messages specific to each asynchronous search technique.

2.2 The Graphical User Interface

S3. Definition and representation of the user interface.

As concerning the interface part, it can be used for the graphical representation of the DisCSP problem's objects (agents, nodes, queens, robots, obstacle, link, etc.) of the patch type. It is recommended to create an initialization procedure for the display surface where the agents' values will be displayed.

To model the surface of the application are used objects of the patches type. Depending on the significance of those agents, they are represented on the NetLogo surface. In Figure 3, 4 is presented the way in NetLogo for representing the agents.



time 0 | edges 18 | Msgs nogood 0 | Msgs ok 43 | Cycles 1 | Constraint checks 31 | Concurrent constraint checks 10

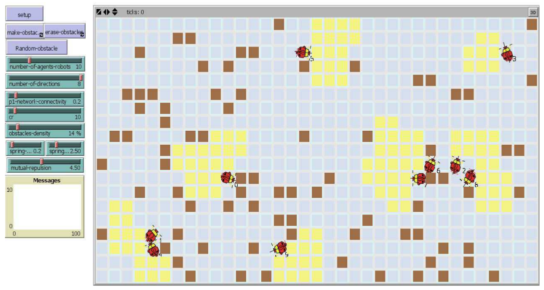
1- setup
2- Layout until it's prett
3- setup-AWCS
4- go-AWCS

The randomly generated (binary) CSPs are characterised by the 4-tuple (n, m,p1,p2),
where:
- n is the number of variables;
- m is the uniform domain size;
- p1 is the portion of the n * (n - 1) /2 possible constraints in the graph;
- p2 is the portion of the m*m value pairs in each constraint that are disallowed by the constraint.

That is, p1 may be thought of as the density of the constraint graph, and p2 as the tightness of constraints.

```
(turtle 9): [["ok?" [4 1 0]] ["ok?" [2 3 0]] ["ok?" [6 2 0]] ["ok?" [6 0 0]] ["ok?" [2 2 0]]]
(turtle 5): [["ok?" [3 2 0]] ["ok?" [0 2 0]] ["ok?" [8 3 0]]]
(turtle 7): [["ok?" [2 3 0]] ["ok?" [0 2 0]] ["ok?" [1 3 0]] ["ok?" [8 3 0]] ["ok?" [2 2 0]]]
(turtle 0): [["ok?" [2 3 0]] ["ok?" [5 2 0]] ["ok?" [7 2 0]] ["ok?" [8 3 0]] ["ok?" [6 2 0]] ["ok?" [6 0 0]] ["ok?" [2 2 0]]]
(turtle 4): [["ok?" [9 0 0]] ["ok?" [2 3 0]] ["ok?" [1 3 0]] ["ok?" [8 3 0]] ["ok?" [2 2 0]]]
(turtle 8): [["ok?" [4 1 0]] ["ok?" [5 2 0]] ["ok?" [7 2 0]] ["ok?" [0 2 0]] ["ok?" [6 2 0]] ["ok?" [6 0 0]]]
(turtle 1): [["ok?" [4 1 0]] ["ok?" [7 2 0]]]
```

(a) Netlogo representation for the graph coloring problem



(b) The 2D square lattice representation for the multi-robot exploration problem

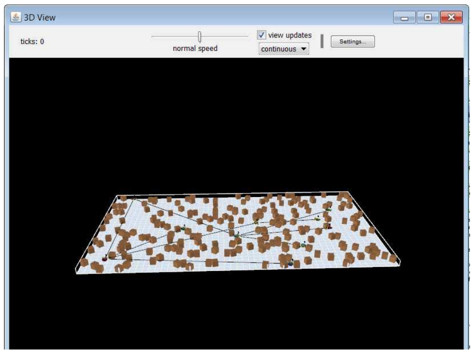
Figure 3. Representation of the environment in the case three problems

S4.Running the DisCSP problems.

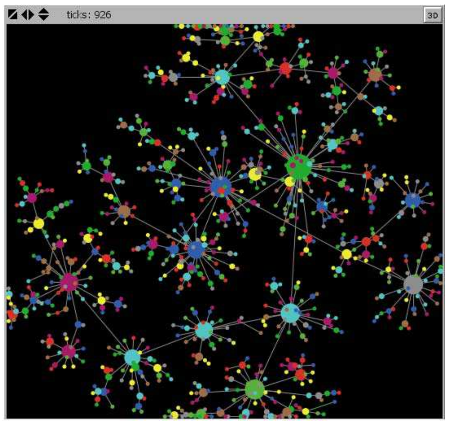
The initialization of the application supposes the building of agents and of the working surface for them. Usually are initialized the working context of the agent, the message queues, the variables that count the effort carried out by the agent. The working surface of the application should contain NetLogo objects through which the parameters of each problem could be controlled in real time: the number of agents (nodes, robots), the density of the constraints graph, etc. These objects allow the definition and monitoring of each problem's parameters. For launching the simulation is proposed the introduction of a graphical object of the button type and setting the forever property. That way, the attached code, in the form of a NetLogo procedure (that is applied on each agent) will run continuously, until emptying the message queues and reaching the stop command. Another important observation is tied to attaching the graphical button to the observer [10]. The use of this approach allows obtaining a solution of implementation with synchronization of the agents' execution. In that case, the observer agent will be the one that will initiate the stopping of the DisCSP algorithm execution (the *go* procedure is attached and handled by the observer). These elements lead to the multiagent system with synchronization of the agents' execution. If it's desired to obtain a system with asynchronous operation, the second method of detection will be used, which supposes another update routine [10], [21]. That new *go* routine will be attached to a graphical object of the button type which is attached and handled by the turtle type agents.

In Figure 5 is captured an implementation of the ABT with temporary links for the random binary problems technique that uses the model presented. The update procedure is attached and handled by the turtle type agents (Figure 5). These elements lead to a multiagent system with agents handling asynchronously the messages. Implementation examples for the ABT family, DisDB, DBS and the AWCS family can be downloaded from the website [21].

More details of implementation of the DisCSP/DCOP in Netlogo are not presented here but are available as a tutorial and downloadable software from [21].

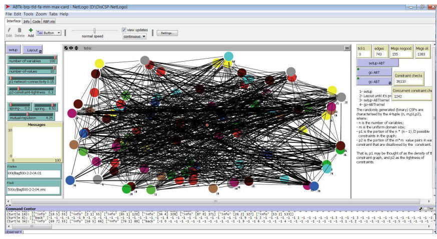


(a) The 3D square lattice representation

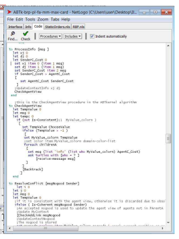


(b) A scale-free network with 900 nodes

Figure 4. Representation of the environment in the case three problems



(a) NetLogo's graphical interface



(b) NetLogo's code tab

Figure 5. NetLogo implementation of the ABT with temporary links for the random binary problems, n=100 agents

2.3 The evaluation of the asynchronous search techniques

Another important thing that can be achieved in NetLogo is related to the evaluation of the asynchronous algorithms. The model presented within this paper allows the monitoring of the various types of metrics:

- the number of messages transmitted during the search: messages-received-ok, messages-received-nogood, messages-received-nogood-obsolete, etc.
- the number of cycles. A cycle consists of the necessary activities that all the agents need in order to read the incoming messages, to execute their local calculations and send messages to the corresponding agents. These metrics allows the evaluation of the global effort for a certain technique
- the number of constraints checked. The time complexity can be also evaluated by using the total number of constraints verified by each agent. It is a measurement of the global time consumed by the agents involved. It allows the evaluation of the local effort of each agent. The number of constraints verified by each agent can be monitored using the variables proprietary to each agents called $nr - constraintc$.
- a number of non-concurrent constraint checks. This can be done by introducing a variable proprietary to each agent, called $AgentC - Cost$. This will hold the number of the constraints concurrent for the agent. This value is sent to the agents to which it is connected. Each agent, when receiving a message that contains a value $SenderC - Cost$, will update its own monitor $AgentC - Cost$ with the new value.
- the total traveled distance by the robots. This metric is specific to the multi-robot exploration problem [3]. It makes it possible to evaluate if an algorithm is effective for mobile and located agents in an unknown environment.

The models presented allow real time visualization of metrics. During runtime, using graphic controls, various metrics are displayed and updated in real time, after each computing cycle. Also, the metrics' evolution can be represented as graphics using plot-like constructions (models from [21] include some templates).

3 Enhancing DisCSP-Netlogo from simulation to real-execution of agents in distributed constraints

3.1 The architecture of the multi-agent system

HubNet is a technology that lets you use NetLogo to run participatory simulations in the classroom. In a participatory simulation [18], a whole class takes part in enacting the behavior of a system as each student controls a part of the system by using an individual device, such as a networked computer or mobile device (tablets and phones with Android, etc). It is based on a client-server architecture, where HubNet hardware includes an up-front computer (server, the "hub") capable of addressing a network of nodes (currently, networked computers, mobile devices with Android or TI-83+ calculators) and a display capability (e.g. computer projection system) enabling an entire class to view the simulation. HubNet enables many users at the "nodes" to control the behavior of individual objects or agents and to view the aggregated results on a joint display.

The software architecture of HubNet [17, 18] contains an NetLogo application called Computer HubNet (HubNet server) and many instances of the client application. Computer HubNet is regarded as a main server and uses networked computers as nodes. The HubNet architecture can support other devices as nodes (mobile device with Android). The network layer implements flexible communication protocols that include the ability to upload and download data sets, support real-time interaction as in network computer games, etc [18]. The activity leader uses the NetLogo application to run a HubNet activity (HubNet server). Participants use a client application to log in and interact with the HubNet server.

Starting from the HubNet architecture we propose a new model that allows modeling and running various search algorithms. The basic idea is to move the agents to run on each network node (computer or mobile device) as opposed to the simulator mode, where they are simulated and run on the same computer. We will adapt this architecture to allow distributed running of the agents in the network. In Fig. 6 is presented the HubNet architecture.

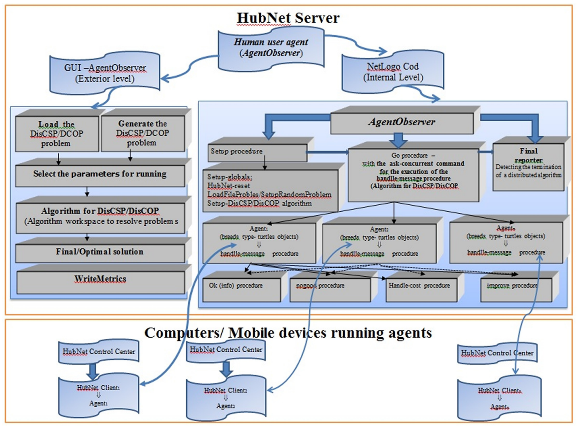

HubNet Server
Human user agent (AgentObserver)
GUI –AgentObserver (Exterior level)
NetLogo Cod (Internal Level)
Load the DisCSP/DCOP problem
Generate the DisCSP/DCOP problem
Select the parameters for running
Algorithm for DisCSP/DisCOP (Algorithm workspace to resolve problems
Final/Optimal solution
WriteMetrics
AgentObserver
Setup procedure
Setup-globals; HubNet-reset LoadFileProbles/SetupRandomProblem Setup-DisCSP/DisCOP algorithm
Go procedure – with the ask-concurrent command for the execution of the handlle-message procedure (Algorithm for DisCSP/DisCOP
Final reporter Detecting the termination of a distributed algorithm
Agent1 (breeds type- turtles objects) ⇓ handle-message procedure
Agent2 (breeds type- turtles objects) ⇓ handle-message procedure
Agentn (breeds type- turtles objects) ⇓ handle-message procedure
Ok (info) procedure
nogood procedure
Handle-cost procedure
improve procedure
Computers/ Mobile devices running agents
HubNet Control Center
HubNet Client1 ⇓ Agent1
HubNet Control Center
HubNet Client2 ⇓ Agent2
HubNet Control Center
HubNet Clientn ⇓ Agentn

Figure 6. The DisCSP architecture for simulating and real-execution of agents in distributed constraints

In this architecture can be remarked the two entities of the distributed application. That is, the central application - HubServer and the client applications - HubNetClient. The central application (HubNet server) contains the definitions of the agents and the procedure for handling the communication chanels. The HubNet clients will be attached to each agent.

3.2 A methodology of implementation and evaluation for the asynchronous search techniques in DisCSP-NetLogo in the real-execution of agents mode

In this paragraph is presented a methodology of implementation for the asynchronous search techniques in NetLogo, using the model presented in the previous paragraph. That methodology supposes the identification of the two entities of the distributed application: the central application - HubServer and the client applications - HubNetClient. Any implementation based on the presented model, will require us to follow the next steps:

S1. Create a NetLogo model according to the previous model for the asynchronous search techniques and for the types of problems used at the evaluation. First, the NetLogo model will require an initialization stage. That includes the interface initialization, initialization of the network module, the activation of HubNet clients and generating the agents. The proposed solution supposes for procedures called *setup*, *login-clients*, *GenerateProblems*. At a minimum it will need the following lines of code in Fig. 7.

S2. Next, all models must also have a *go* (update) procedure. The wrapper runs the NetLogo program by asking it to loop for a certain number of times and allows the finalizing of the DisCSP algorithm.

```
to setup ; Setup the model for a run, build a constraints graph.
  setup-globals ; setup Global Variables
  setup-patches ; initialize the surface of the application
  are used objects of the patches type
  hubnet-reset; we initialize the network mode, which will ask
  ;the user for a session name and
  ;open up the HubNet Control Center
  ...
end
```

(a) The Setup Procedure in DisCSP-Netlogo

```
to login-clients ; allows agents-HubNet Client to log into the
;activity without running the model or collecting data
  while [ hubnet-message-waiting? ] [
  hubnet-fetch-message ;get the first message in the queue
  ifelse hubnet-enter-message?; The clients send messages when it login
  [ create-new-agents
  hubnet-send agent-id "Accept-agent" "Yes"]
  [ ..]
  ]end
```

(b) The login procedure

```
to GenerateProblems ; Build a constraints graph.
  setup-random-problem ; we generate the types of problems
used at the evaluation.
  CalculateOrders; is selected from variable-ordering heuristics
  ...
end
```

(c) The GenerateProblems Procedure in DisCSP-Netlogo

Figure 7. Initialization of the multi agent system.

Usually for the DisCSP algorithms, the solution is generally detected only after a break period in sending messages (this means there is no message being transmitted, state called quiescence). In such a procedure, that needs to run continuously (until emptying the message queues) for each agent, the message queue is verified (to detect a possible break in message transmission). In the case of the extended solution in which the agents run as HubNet clients, the checking is done by each agent, but the informations are transmited to the central application (HubNet server) which decides (in the case that all the queues are empty) that a state called quiescence is reached.

Sample code for the *go* procedure in the case of asynchronous search techniques can be found in Fig. 8.

Another observation, each HubNet client signals the reception of a message and transmits to the HubNet server application that thing, for the latter to run the agent's code. The solution with HubNet clients (of the NetLogo type) doesn't allow running the code effectively by the clients. In a ulterior version we will extend the HubNet clients functionality so that they will run entirely the code (using Java modules that communicate using sockets).

Another observation, the HubNet clients don't allow direct transmission (peer-to-peer) of the messages. They are brokered by the central HubNet server application.

The procedure should also allow the management of messages that are transmitted by the agents. For that, when a HubNet client receives a message, it needs to call another procedure (that is called handle-message) and is used to handle messages specific to each asynchronous search technique. The handling of the communication channels will be performed by this central agent. These elements will lead to a variant of implementation in which the synchronizing of the agents' execution is done.

The distributed application's work flow is composed of the following steps:

```
to go // The running procedure
 every 0.1 [
    set no-more-messages true
    ;get commands and data from the clients
listen-to-clients -as long as there are more messages from the clients keep processing the
    while [ hubnet-message-waiting? ] [
    hubnet-fetch-message ;get the first message in the queue
    ifelse hubnet-enter-message?; The clients send messages when it login
    [ hubnet-send hubnet-message-source "Accept-agent" "No" ]
    [ if hubnet-exit-message?; The clients send messages when it logout
      [Show "Error - too few agents " : stop ]
      [Process-Queue hubnet-message-tag ]
    ]
    ask-concurrent agents [
      if (not empty? message-queue)[ set no-more-messages false]
  ]
  ifelse (no-more-messages and Not done)
    [WriteSolution : hubnet-broadcast "Solution "Yes" : stop]
    [if (done)
      [show "No solution" : hubnet-broadcast "Solution" "No solution" :stop]
end
```

Figure 8. The Go Procedure in DisCSP-Netlogo for the asynchronous search techniques with synchronization of the agents' execution

S1. Start the HubNet Server. It is done by pressing the initialization button *setup*. The runtime parameters are set: number of variables, their domains, the constraints graph density (p1-network-connectivity), etc. The button will initialize the network support.

S2. Activating the connections with the clients. For that press the *login* button on the computer with the HubNet server to allow the clients to connect. On each client computer is launched the clients manager (*HubNet Control Center*). Using that tool the HubNet Clients are opened, a username is chosen and connect to the main activity (to HubNet Server). When all the HubNet clients have connected the *login* button is disabled.

S3. An instance for the evaluated problem is generated. For that the button *GenerateProblems* is pressed. Optional, the *Layout* button can be used, to redraw the surface on the screen, until we consider it to be pretty.

S4. The button *setup-DisCSP* is activated. That will initialize the data structures necessary for the DisCSP/DCOP algorithm.

S5. The main application is run (on the HubServer) using the *go* button.

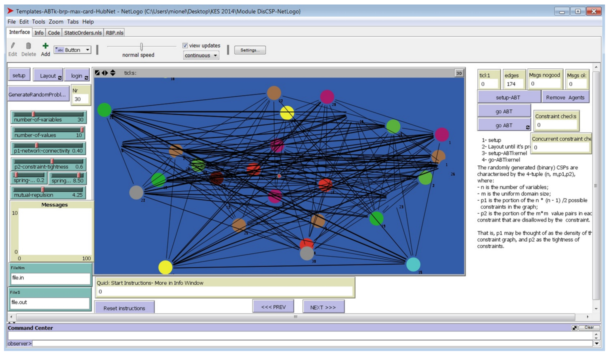
S6. Finally, the measurements and the problem's solution are collected.

In Fig. 9 are presented two captures for the two entities, HubNet-Server and HubNetClient.

4 Running on a Linux cluster

In this paragraph we will present a methodology to run the proposed NetLogo models in a cluster computing environment or on a single machine. We utilize the Java API of NetLogo as well as LoadLeveler. LoadLeveler is a job scheduler written by IBM, to control scheduling of batch jobs. This solution is not restricted to operate only in this configuration, it can be used on any cluster with Java support and it operates with other job schedulers as well (such as Condor).

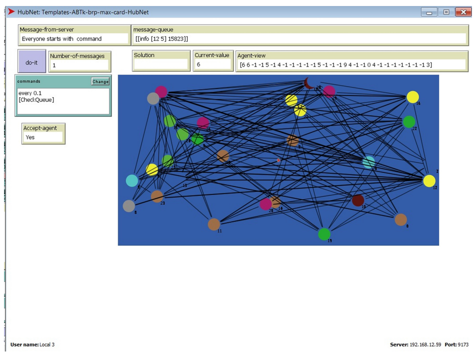
Such a solution will allow running a large number of agents (nodes, variables, robots, queens, etc.). The first tests allowed running of as much as 500 agents, in the conditions of a high density constraint graph. The first experiments were done on the InfraGrid cluster from the UVT HPC Centre [23], on 100 computing systems (an hybrid x86 and NVIDIA Tesla based). InfraGRID is an Linux only cluster based on a mixture of RedHat Enterprise Linux 6 and


1- setup
2- Layout until it's pr
3- setup-ABTkernel
4- go-ABTkernel

The randomly generated (binary) CSPs are characterised by the 4-tuple (n, m,p1,p2), where:
- n is the number of variables;
- m is the uniform domain size;
- p1 is the portion of the n * (n - 1) /2 possible constraints in the graph;
- p2 is the portion of the m*m value pairs in eac constraint that are disallowed by the constraint.

That is, p1 may be thought of as the density of th constraint graph, and p2 as the tightness of constraints.

(a) The HubNetServer



(b) The HubNetClient

Figure 9. NetLogo implementation of the ABT -HubNet version

CentOS 6. For Workload Management, JOB execution is managed at the lowest level by IBM LoadLeveler.

The methodology proposed in the previous paragraph that uses the GUI interface will run on a single computer. In this paragraph we will present a new solution, without GUI, that can run on a single computer or on a cluster.

The proposed approach uses the NetLogo model presented previously, runnable without the GUI, with many modifications. In order to run the model in that manner is used a tool named BehaviorSpace, existent in NetLogo. BehaviorSpace is a software tool integrated with NetLogo that allows you to perform experiments with models in the "headless" mode, that is, from the command line, without any graphical user interface (GUI). This is useful for automating runs on a single machine, and can also be used for running on a cluster.

BehaviorSpace runs a model many times, systematically varying the model's settings and recording the results of each model's run. Using this tool we develop an experiment that can be runned on a single computer (with a small number of agents) or, in the headless mode on a cluster (with a large number of agents).

We will now present the methodology for creating such an experiment. The steps necessary for the implementation of a multiagent system are as follows:

S1. Create a NetLogo model according to the previous model for the asynchronous search techniques and for the types of problems used at the evaluation. For running it on the cluster and without GUI some adaptations have to be made. First, the NetLogo model must have a procedure called setup to instantiate the model and to prepare the output files. At a minimum it will need the following lines of code in Figure 10.

```
to setup ; Setup the model for a run, build a constraints graph.
  setup-globals ; setup Global Variables
  setup-patches ; initialize the work surface on which the agents move
  setup-turtles
; we generate the objects of the turtles type that simulate the agents
  setup-random-problem
; we generate the types of problems used at the evaluation.
  setup-DisCSP
; we initialize the data structures necessary for the DisCSP algorithm
end
```

Figure 10. The Setup Procedure in DisCSP-Netlogo.

Next, all models must also have a *go* (update) procedure. In such a procedure, that needs to run continuously (until emptying the message queues) for each agent, the message queue is verified (to detect a possible break in message transmitting).

The procedure should also allow the management of messages that are transmitted by the agents. The procedure needs to call for each agent another procedure (that is called handle-message) and is used to handle messages specific to each asynchronous search technique.

```
to go // The running procedure
   set no-more-messages true
   set nr-cycles nr-cycles + 1
   ask-concurrent agents [
     if (not empty? message-queue)[
       set no-more-messages false]]
   if (no-more-messages) [WriteSolution
   stop]
   ask-concurrent agents [handle-message]
end
```

Figure 11. The Go Procedure in DisCSP-Netlogo for the asynchronous search techniques with synchronization of the agents' execution

The first solution of termination detection is based on some of the facilities of the NetLogo: the ask-concurrent command that allows the execution of the computations for each agent and the existence of the central observer agent. The handling of the communication channels will be performed by this central agent. These elements will lead to a variant of implementation in which the synchronizing of the agents' execution is done. Sample code for the *go* procedure in the case of asynchronous search techniques can be found in Figure 11.

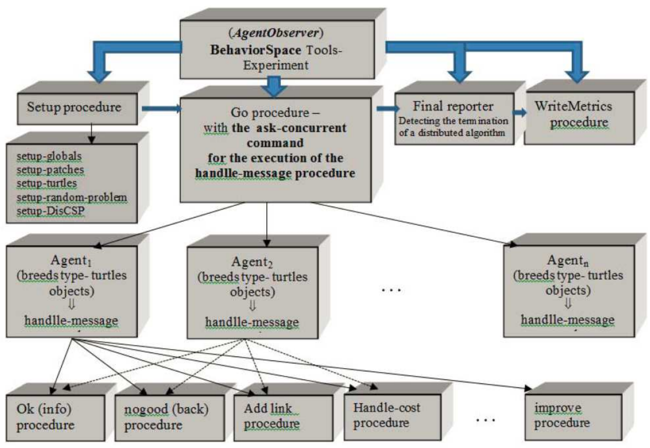
S2. Create an experiment using BehaviorSpace and parse the NetLogo file into an input XML file (so that it can be runned in the headless mode, that is without GUI). In Figure 12 is presented a simple example of XML file.

```
<experiments>
   <experiment name="experiment" repetitions="10" >
   <setup>setup< /setup>
   <go>go-mrp< /go>
   <final>WriteMetrics< /final>
   <exitCondition>Final< /exitCondition>
   <enumeratedValueSet variable="p1-network-connectivity">
     <value value="0.2"/>
   ...
   < /experiment>
< /experiments>
```

Figure 12. The XML file for the multi-robot exploration problem

To finalize the run and adding up the results it is recommended the use of a Netlogo reporter and a routine that writes the results. The run stops if this reporter becomes true.

S3. Create a Linux shell script (in sh or in bash) that describes the job for LoadLeveler. Once the Netlogo model is completed with the experiment created with the BehaviorSpace tool, it is time to prepare the system for multiple runs.



(AgentObserver) BehaviorSpace Tools-Experiment
Setup procedure
setup-globals
setup-patches
setup-turtles
setup-random-problem
setup-DisCSP
Go procedure – with the ask-concurrent command for the execution of the handlle-message procedure
Final reporter
Detecting the termination of a distributed algorithm
WriteMetrics procedure
Agent1 (breeds type- turtles objects) handlle-message
Agent2 (breeds type- turtles objects) handlle-message
Agentn (breeds type- turtles objects) handlle-message
Ok (info) procedure
nogood (back) procedure
Add link procedure
Handle-cost procedure
improve procedure

Figure 13. Architecture of a multiagent system with synchronization of the agents' execution

In Figure 13 is presented this multiagent system's architecture for running on a cluster of computers.

5 Discussion

There are very few platforms for implementing and solving DisCSP problems: DisChoco [19], DCOPolis [13] and FRODO [5]. In [19] a DisCSP/DCOP platform should have the following features:

- be reliable and modular, so it is easy to personalize and extend;
- be independent from the communication system;
- allow the simulation of multiagent systems on a single machine;
- make it easy to implement a real distributed framework;
- allow the design of agents with local constraint networks.

It is interesting to see if the proposed platform brings some benefits compared to other platforms. The solution presented in this paper, based on NetLogo, has these features:

- the modules can be adapted and personalised for each algorithm. There is a very large community of NetLogo users that can help for development.
- it allows the communication between agents, without being necessary to call directly the communication system (it is independent of the network support);
- the models can allow the simulation of multiagent systems on a single machine, and also on a cluster;
- DisCSP-NetLogo provides a special agent *Observer*, that is responsible for simulation initialisation and control interface. The AgentObserver allows the user to track operations of a DisCSP algorithm during its execution. Also, there are 4 tools (Globals Monitor, Turtle Monitor, Patch Monitor and Link Monitor) that allow monitoring of global variables values, the values of the variables associated to the agents.
- there are facilities such as *agentsets* that allow the implementation of agents that manage more variables.
- manipulating large quantities of information requires the use of databases, for example for nogood management, using the SQL extension of NetLogo we can store and access values from databases.
- NetLogo allows users to write new commands in Java and use them in their models (using extensions).

Another discussion refers to the advantages of running NetLogo models on a cluster of computers in the variants with complete NetLogo or with minimal install (core branch).

In most of the articles about DisCSP/DCOP, the evaluations of the algorithms are made for maximum 100 agents. The cluster allowed running instances over 500 agents, with various difficulties (even 1000 but with a lower difficulty)[10]. It is interesting to see what is the effect of running the models on a cluster of computers, if the runtime is reduced for the analyzed techniques. For that, we performed some empirical tests with a variable number of agents (n=500 and n=1000). We examined the performance of AWCS in scale-free networks (we implemented and generated in NetLogo both solvable and unsolvable problems that have a structure of scale-free networks). Scale-free networks are generated with the following parameters: nodes = 500, $|D_i|$ = 10, md = 4 and γ = 1.8, respectively nodes = 1000, md = 4 and γ = 2.1.

For the evaluations, we generate five scale-free networks. For each network, the constraint tightness is fixed at 0.4 and 100 random problem instances are generated. The runs were performed in three variants: on a single computer using the model with GUI (C1), on a single computer, but without GUI (headless (C2)) and on the Infragrid cluster (C3). For each was counted the total amount of time for running 100 instances. The results are the folowing: Nodes=**1000**, C1–**8h,53min**, C2–**3h,8min** and C3–**38min**, respectively Nodes=**500**, C1–**47min**, C2–**16min** and C3–**5min**.

The analysis of the results shows that the solution running on a cluster of computers allows problems with big dimensions for the addressed problems, and also a short runtime. The platform based on NetLogo has the following differences with the other platforms:

1. The sources of the algorithms are accessible, one can obtain implementations derived from the DisCSP/DCOP algorithms.
2. The programmer has complete access to the algorithms code and can intervene.
3. The possibility to run on a cluster of computers allows the simulation of problems with thousand or tens of thousands agents. In most of the studies performed, one can remark that most of the tests are done on problems with a small number of agents (below 100).

In Table 1 are presented the difference between platforms for implementing and solving DisCSP/DCOP problems.

6 CONCLUSION

In this paper we introduce an model of study and evaluation for the asynchronous search techniques in NetLogo using the typical problems used for evaluation, model called DisCSP-NetLogo.

An open-source solution for implementation and evaluation of the asynchronous search techniques in NetLogo, for a great number of agents, model that can be run on a cluster of computers is presented. Such a tool allows the use of various search techniques so that we can decide on the most suitable one.

In this paper we presented a methodology to run NetLogo models in a cluster computing environment or on a single machine, varying

Table 1. The difference between platforms for implementing and solving DisCSP/DCOP problems

The platform	Support for implementing the DisCSP algorithms	Support for implementing DCOP algorithms	Support for real running in an distributed environment	Access to algorithm sources	Implementation of most of the DisCSP algorithms	Implementation of most of the DCOP algorithms	Being able to run on a cluster
DisCo	Yes	Yes	No	No	Yes	Yes	No
DisChoco	Yes	Yes	Yes	Yes	partial	partial	Yes
DCOPolis	No	Yes	Yes	No	No	Yes	?
FRODO	No	Yes	Yes	No	No	Yes	Yes
DiCSP-Netlogo	Yes	Yes	Yes	Yes	Yes	partial	Yes

both parameter values and/or random number of agents. We utilize the Java API of NetLogo as well as LoadLeveler. The solution without GUI allows to be run on a cluster of computers in the mode with synchronization, as opposed to the GUI solution that can be run on a single computer and allows running in both ways: with synchronization or completely asynchronously.

The open-source solution presented in this paper can be used as an alternative for testing the asynchronous search techniques, in parallel with the platforms already consecrated as DisChoco, DCOPolis, FRODO, etc. A comparison with the main evaluation and testing platforms for distributed constraints search and optimization algorithms is presented.

As future developments, we will try to implement the modules on the HPC Repast platform, in a Logo-like C++. This thing will allow running on supercomputers with a very high number of agents (above 100,000 agents).

REFERENCES

[1] A. L. Barabasi and A. L Albert, *Emergence of scaling in random networks*, Science, 286 (1999), p. 509-512.

[2] C. Bessiere, I. Brito, A. Maestre, P. Meseguer. *Asynchronous Backtracking without Adding Links: A New Member in the ABT Family*. A.I., 161:7-24, 2005.

[3] A. Doniec, N. Bouraqadi, M. Defoort, V-T Le, and S. Stinckwich. *Multi-robot exploration under communication constraint: a disCSP approach*. In 5th National Conference on "Control Architecture of Robots", May 18-19, 2010 - Douai, FRANCE.

[4] A. Grubshtein, N. Herschhorn, A. Netzer, G. Rapaport, G. Yafe and A. Meisels. *The Distributed Constraints (DisCo) Simulation Tool*. Proceedings of the IJCAI11 Workshop on Distributed Constraint Reasoning (DCR11), pages 30–42, Barcelona, 2011.

[5] Leaute, T., Ottens, B., Szymanek, R.: *FRODO 2.0: An open-source framework for distributed constraint optimization*. In Proceedings of the IJCAI'09 Distributed Constraint Reasoning Workshop (DCR'09). pp. 160–164. Pasadena, California, USA, 2009. Available: http://liawww.ep.ch/frodo/.

[6] Lytinen, S. L. and Railsback, S. F. *The evolution of agent-based simulation platforms: A review of NetLogo 5.0 and ReLogo*. In Proceedings of the Fourth International Symposium on Agent-Based Modeling and Simulation, Vienna, 2012.

[7] A. Meisels, *Distributed Search by Constrained Agents: algorithms, performance, communication*, Springer Verlag, London, 2008.

[8] Modi, P., Shen, W.-M., Tambe, M., Yokoo, M., *ADOPT: Asynchronous distributed constraint optimization with quality guarantees*, A.I., 2005, 161 (1-2), pp. 149-180.

[9] Monier, P., Piechowiak, S. et Mandiau, R. *A complete algorithm for DisCSP: Distributed Backtracking with Sessions (DBS)*. In Second International Workshop on: Optimisation in Multi-Agent Systems (OptMas), Eigth Joint Conference on Autonomous and Multi-Agent Systems (AAMAS 2009), Budapest, Hungary, pp 3946.

[10] Muscalagiu I, Popa HE, Vidal J. *Large Scale Multi-Agent-Based Simulation using NetLogo for implementation and evaluation of the distributed constraints. In proceedings of IJCAI DCR 2013 (23rd International Joint Conference on Artificial Intelligence - Workshop on Distributed Constraint Reasoning*, 2013, Beijing, August, pp. 60 - 74.

[11] Muscalagiu I, Popa HE, Vidal J. *Enhancing DisCSP-Netlogo from simulation to real-execution of agents in distributed constraints. In proceedings of 18th Annual International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES 2014)*, Procedia Computer Science 35 (2014) 261 270.

[12] A. Petcu. *A Class of Algorithms for Distributed Constraint Optimization*. PhD. Thesis No. 3942, Swiss Federal Institute of Technology (EPFL), Lausanne, 2007.

[13] Sultanik, E.A., Lass, R.N., Regli,W.C. *Dcopolis: a framework for simulating and deploying distributed constraint reasoning algorithms*. AAMAS Demos, page 1667-1668. IFAAMAS, (2008).

[14] Silaghi M.C., D. Sam-Haroud, B. Faltings. *Asynchronous Search with Aggregations*. In Proceedings AAAI'00, 917-922.

[15] S. Tisue and U. Wilensky. *Netlogo: Design and implementation of a multi-agent modeling environment*. In Proceedings of the Agent 2004 Conference, 2004, pp 7 - 9.

[16] M. Yokoo, E. H. Durfee, T. Ishida, K. Kuwabara. *The distributed constraint satisfaction problem: formalization and algorithms*. IEEE Transactions on Knowledge and Data Engineering 10(5), page. 673-685, 1998.

[17] U. Wilensky. *NetLogo itself: NetLogo*. Available: http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, 1999.

[18] Wilensky U, Stroup W. *HubNet*. Center for Connected Learning and Computer-Based Modeling, Northwestern University: Evanston, IL USA, ¡http://ccl.northwestern.edu/ps/¿, 1999.

[19] M. Wahbi, R. Ezzahir, C. Bessiere, El Houssine Bouyakhf. *DisChoco 2: A Platform for Distributed Constraint Reasoning*. Proceedings of the IJCAI11 Workshop on Distributed Constraint Reasoning (DCR11), pages 112–121, Barcelona, 2011.

[20] J. Vidal. *Fundamentals of Multiagent Systems with NetLogo Examples*. Available: http://multiagent.com/p/fundamentals-of-multiagent-systems.html.

[21] *MAS NetLogo Models-a*. Available: http://discsp-netlogo.fih.upt.ro/.

[22] *MAS NetLogo Models-b*. Available:http://jmvidal.cse.sc.edu/netlogomas/.

[23] *InfraGRID Cluster*. Available:http://hpc.uvt.ro/infrastructure/infragrid/

:

The Post-Modern Homunculus

Eric Neufeld and Sonje Finnestad[1]

Abstract. Throughout the ages, magicians, scientists and charlatans have created life-*like* artifacts purported, in some cases, to be intelligent. In their day, these artifacts were remarkable, but the question arises as to whether they could really think. In one famous case, *the Chess Player,* the intelligence was literally a little person, hidden inside the machine, doing the intelligent work. Analogously, throughout the history of philosophy, and cognition, there have been theories to explain intelligence in humans, how it works, whether it can be replicated or imitated with various artifacts, and how we then evaluate our progress towards achieving intelligence with such artifacts once created? A philosophical problem with many such theories is that they amount to what is called a *homunculus* argument – the account, upon close scrutiny reveals a "little one" (homunculus) somewhere in the mind that is responsible for thinking, thereby regressing the problem a step, rather than answering it. For most of the computing era, the Imitation Game put forward by Alan Turing has been considered the gold standard for deciding intelligence, though recently Hector Levesque has pointedly argued that the time has come to abandon Turing's test for a better one of his own design, which he describes in a series of acclaimed papers. In particular, we argue that Levesque, who has cleverly found the 'homunculus' in the arguments of others, has essentially regressed the problem of intelligence to a homunculus in his own system.

1 INTRODUCTION

In 18th century Europe, many people were fascinated by the *living anatomies* created by Jacques de Vaucanson. His first creation was a life-sized flute player that played 12 songs on a real transverse flute, a challenging task for any human, without significant practice and training. The French Academy of Science took it seriously and concluded that: *this machine was extremely ingenious; that the creator must have employed simple and new means, both to give the necessary movements to the fingers of this figure and to modify the wind that enters the flute by increasing or diminishing the speed according to the different sounds, by varying the position of the lips, by moving a valve which gives the functions of a tongue, and, at last, by imitating with art all that the human being is obliged to do.* [history-computer.com, 2016]

Another automaton played 20 different tunes on a flute and – with the other hand – the tambourine and, apparently, played them well. But most marvelous of all was *The Digesting Duck*. The duck, which had more than 400 moving parts (perhaps over 400 in each wing – opinions differ) could flap its wings, drink, eat, digest grain, and defecate, all in a remarkably realistic manner.

All its movements, external and internal, were copied from nature. There were, of course, limits to what could be achieved: "I do not claim that this digestion is a perfect digestion, able to make blood and nourishing particles to nurture the animal; to reproach me for this, I think, would show bad grace." Nonetheless, "attentive people will understand the difficulty to make my automaton perform so many different movements." None of these automata would have been mistaken by humans for real, live entities, human or avian, and that was never the intent: they were truly remarkable mechanical imitations of life.

Another notable automaton – though for rather different reasons – was *Maelzel's Chess-Machine*, created by Baron von Kempelen and improved by Johann Nepomuk Maelzel [3]. Unlike Vaucanson, Maelzel made fantastic claims for his automaton: he claimed that it could actually play chess. And indeed it did play a strong game of chess, beating – among others – Benjamin Franklin – but was eventually exposed as a hoax by Edgar Allen Poe, who was certain that no mere machine could have the intelligence necessary to play chess. Although mechanically ingenious as regards how it moved the pieces, the artifact contained a concealed human chess player that actually decided the moves. Inside the mechanism, there was a *homunculus*!

Why tell this curious tale? Creation of artificial intelligence has been one of humanity's long-standing dreams, and many believe it to be within the realm of possibility – so much so, that many have been willing to believe the hoaxes perpetuated by the likes of Maelzel for some time. In more modern times, the homunculus slips in benignly, or unintentionally. That is, theories arise in philosophy and cognition, and it takes some scrutiny to determine or detect the presence of a homunculus. For instance, a theory of human vision might note that the human eye works much like a camera, with the lens projecting an upside-down image of the world on the retina, which some internal mechanism in the brain can trivially watch and interpret. This example is simple, but the "internal mechanism" is a homunculus that has solved the problem we proposed to explain. (This is an oversimplified presentation of the Cartesian Theatre [3].) Sometimes homunculus arguments are similar to regression arguments.

In a series of articles, several authors, notably Hector Levesque, revisit key foundational questions in artificial intelligence. Levesque, although he does not use the term *homunculus,* brilliantly uncovers one [7] in the celebrated Chinese Room thought experiment of John Searle. (We disagree somewhat with his explanation, but more on that later. His 'reveal' is nonetheless astonishing.)

Here's the rub. In other papers [8,9], Levesque goes on to explore foundational questions in Artificial Intelligence – whether behavior is sufficient or necessary and at one point asks whether the Turing Test is obsolete and should be replaced by something

[1] Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, Canada, email: eric.neufeld@usask.ca

else, and poses a very clever constructive alternative. His claim was startling enough to make the *New Yorker*, a significant achievement. However, we argue that that, one, Levesque's new test also uses (inadvertently) a homunculus argument (although it was very hard to find), and two, that the Turing test is of a fundamentally different character than many of the other landmark tests that have come and gone in AI. In fact, in light of the special theme of this conference on *AI and Human Values*, we make the strong claim that Turing's test transcends formal science, whereas most of the other artificial intelligence tests are benchmarks or milestones. Other kinds of judgments also transcend [or something like that?] also transcends science, and so these considerations intersect with this central theme of human values.

The following elaborates these points.

2 INTRODUCTION

2.1 The paradox of mechanical reason

John Haugeland [4], in his entertaining textbook, discusses what he describes as "The Paradox of Mechanical Reason." (We chose this reference because it is an accessible to both computer scientists and philosophers.) The paradox of mechanical reason is associated with the 'computational model' of reason as "the manipulation of meaningful symbols according to rational rules (in an integrated system)". (In the heyday of knowledge representation, this was sometimes called the *physical symbol system* hypothesis [11].)

The puzzle is this: if the manipulator of the symbols pays attention to their meaning, can it be entirely mechanical? After all, it somehow "knows" what the symbols *mean*. But if it doesn't pay attention to meanings, can it truly reason? To put it in a computational setting, imagine a machine generating sentences by putting symbols together. If the machine pays no attention to the meaning, won't most of the output be rubbish, and won't any sensible prose be the results of chance? Or – if the sentences are quite engaging, must it be that there is something more than a machine producing them? Hence the paradox: "if a process or system is mechanical, it can't reason; if it reasons, it can't be mechanical."

Note how we have hedged our language. We don't at this time wish to altogether rule out the possibility of building a successful mechanical reasoner. Perhaps *we*, at this point in time, are insufficiently skilled to see our way through to the solution, but one day someone may find that it is possible to build a mechanical reasoner with a handy built-in symbol manipulator that does a terrific job. Now the question arises: how does this built-in symbol manipulator work?

The argument regresses. Either the *manipulator* pays attention to the meaning of the symbols, or it does not. If the manipulator pays attention to 'meaning', how does it do what it does? And, to the contrary, if it doesn't, how does it accomplish what it does? Now we must explain the workings of the manipulator. It's a *homunculus* – a hidden version of what we are purporting to study tucked in, usually unnoticed. Since we are "explaining intelligence by presupposing it" - an endless regress of explanatory homunculi seems unavoidable.

Haugeland argues that we are not stuck with an infinite regress if the manipulator-homunculus can be recursively decomposed into progressively smaller and 'dimmer' homunculi until "the last have no intelligence at all." But, to the best of our knowledge, this strategy has not resulted in any notable successes, though one might argue it bears some resemblance to neural nets and/or deep learning (which have yet to pass the Turing Test). Nor have we found a result that handles just the very last step – taking the dimmest of the homunculi and powering them by unintelligent machines.

We return to Haugeland's treatment of the paradox. Here, his 'fundamental strategy' is *redescription* or *interpretation*, which is "redescription grounded in the coherence of a text. (Truth is part of this, but not the whole.) The two sides of the paradox, according to Haugeland, represent "two different modes of description. From one point of view, the inner players are mere automatic formal systems, manipulating certain tokens in a manner that accords with certain rules, all quite mechanically. But from another point of view, those very same players manipulate those very same tokens – now interpreted as symbols – in a manner that accords quite reasonably with what they mean."

This still leaves us with what Haugeland calls *the mystery of original meaning*. Haugeland accepts that there is a difference between what he calls *original meaning* that is 'already there' in what is interpreted and *derivative meaning* that derives (only) from interpretation. "The questions remain: Which symbolic systems have their meanings originally (non-derivatively) and why?"

He suggests that the answer may have something to do with what he calls *semantic activity*. Semantically active symbols "interact and change" in ways that are "semantically appropriate". They are active in ways that make sense. Nonetheless, the matter is complicated. A calculator is semantically active but 'in some intuitively compelling sense, it "has no idea" what numbers are'. (But see below; Levesque may disagree.) On the other hand, when it comes to exceedingly sophisticated robots, possessing 'mobile and versatile bodies' and 'capable (to all appearances anyway) of the full range of "human" communication, problem solving, artistry, heroism, and what have you', if such things ever exist, "then by all means, they can have original meaning."

Let's rephrase this discussion of semantics in the language commonly used in the Knowledge Representation community, especially the logicist group.

First, let us begin with a set of formal marks – a straightforward language of characters and symbols. These can be alphanumeric characters and numerals, as well as marks interpreted as logical or relational operators, and so on. Of course, the best example of this is natural language, but AIers, especially those in the Knowledge Representation community, have hoped for something where the core symbols are more rigidly defined, and vagueness and imprecision are incorporated by adding rather precise formalisms such as error bounds and probabilities that allow the expression of uncertainty though they quantify the amount and nature of the certainty quite exactly.

Suppose also, that on the mechanical side, we develop a *syntactic* machine that can manipulate these symbols according to a set of fixed rules that pays no attention to their meaning. And suppose finally we are quite good at tuning this machine so that it gives us the kinds of answers we want. (In fact, by looking at the work in logic programming and automated theorem proving, it seems we actually *are* quite good at this.)

On the reasoning side, we develop a *semantics* (set of meanings) for the formal marks, that is, a mapping from the formal marks into objects and relationships in real or artificial worlds about which we wish to reason, that tells us humans how to interpret the formal marks, and how to interpret new combinations of the formal marks that might arise, and furthermore, this process of interpretation of the formal marks never, ever yields an untruth, once the meanings are defined.

Because we are so clever at tuning the syntactic machine to give us the answers we want, suppose we get to a point where we develop an actual meta-proof that shows that the *semantical* side is exactly equivalent to the *syntactical* side, in this sense: for every truth produced via the semantical rules, there is a way to derive the same sentence on the syntactical machine, *and vice-versa.* The formal characterization of this is that our language of formal marks is *sound* and *complete*. Moreover, what we have presented as a kind of thought experiment actually exists: the first order predicate calculus has this property; it has a formal semantics and a simple syntax (including rules of proof), such that anything true vis-à-vis the semantics, can also be can be derived by the machine, which knows not what it says. And whenever a machine derives a sentence of symbols, it is true in the semantical sense.

Thus spake Tarski: "Snow is white if 'snow is white'." The machine can derive anything that is true in the real world, and anything the machine derives is true in the real world. This, surely, was a revolutionary advance for the possibilities of mechanical reason! It certainly was true in mathematics, and this belief was widely held in the early days of "logic-based AI", and even revived somewhat because of the computational efficiency and simplicity of the resolution rule.

We return to this paradox shortly. We first need to give an example of a homunculus argument, so we use Levesque's critique of Searle.

2.2 Searle's Chinese Room

Searle's Chinese Room [12] offers a different take on the paradox of mechanical reasoning. Searle wants to create a divide between outward behavior and true intelligence – whatever that means. "Imagine," he says, "a native English speaker who knows no Chinese locked in a room full of boxes of Chinese symbols (a data base) together with a book of instructions for manipulating the symbols (the program)." People outside the room pass in questions to the English speaker, who mechanically (!) consults his instruction book, retrieves certain answer symbols from the boxes, and returns them to the asker.

Suppose further that the asker is pretty happy with the answers. The question asked when Searle's puzzle is discussed, especially in AI circles, is "Is the behavior of the English speaker *intelligent*?"

Responses to this question can be divided into two major camps. The first camp argues that his or her behavior is intelligent. If we can build the computational equivalent of Searle's Chinese Room to solve other real world problems, generally getting good answers, what do we care whether it understands what it is doing? We have a system that appears, for all intents and purposes, to be intelligent, and we can trust its answers. This seems obvious.

The second camp argues that the English speaker's behavior is *not* intelligent, because the English speaker has no idea as to what is being discussed, even if he or she is a fully capable human. This also seems obvious.

From the point of view of constructing intelligent computers, both answers have some merit. The first camp puts pragmatics first; the second camp wishes to bring a deeper notion of understanding into the argument.

The question can also be asked in a different way. "Is the behavior of the *system* intelligent?" Putting the question this way, the system includes the English speaker, the Chinese Room full of boxes, and the instruction manual.

This question is much tougher. The English speaker is reduced to a cog in a more complex system – it has been reduced to one of Haugeland's dimmest (or even most mechanical) homunculi, but the whole seems to be greater than the sum of the parts, which includes the all-important instruction manual.

Thirty years later, Searle [13] restated the point of the argument: "Computation is defined purely formally or syntactically, whereas minds have actual mental or semantic contents, and we cannot get from syntactical to the semantic just by having the syntactical operations and nothing else." In other words, "no". Searle's view is that neither the person nor the system is intelligent, because however clever the answers of the system are, the system does not understand what it is doing.

We do not intend to discuss the many responses to Searle's argument or even the question of its validity, because we believe, as Levesque [7] baldly states, *there is no such instruction book* – and the question as posed is moot.

Levesque writes that "Searle exploits the fact that we do not yet have a clear picture of what a real book for Chinese would have to be like." Why? Because such a book would have to have an answer for every possible question asked in every possible context.

In other words, the *instruction book* is a homunculus!

For Searle's Chinese Room to function as described, someone would have had already solved the problem of mechanical reason perfectly and incorporated it into the instruction book, whether this takes the form of paper, a hard drive, a computer, a database, or a distributed system.

This homunculus argument is a little different from the Cartesian Theatre. Searle doesn't use the homunculus to explain a phenomenon, but its presence in the story renders the argument somewhat moot: If we choose to agree that the instruction manual exists, Searle's argument can stand, but if we believe it cannot exist without solving the AI problem, then there is not much point to continuing the discussion.

Now – at this point, we diverge significantly from Levesque, who takes a different tack. After saying he wishes to argue there are no such instruction books for Chinese, Levesque states that proving such a point will be a challenge. "All we can do is wave our hands." He points out the variety and complexity of the interpretations of the Chinese Room Problem. Instead, Levesque uses an analogous argument, which he calls the Summation Room.

The Summation Room, as we might expect, contains a person with no understanding of addition. The input to the analogical "English speaker" is a list of numbers, and the output is the sum of these numbers. The instruction manual contains the answers to all possible sums. (Levesque restricts the size of the numbers so that the book is finite.) Levesque argues that such a book, one that would enable a person to produce the correct answer to an addition question without actually performing addition, cannot exist, because the new instruction manual would have to be bigger than the known universe. (We are omitting some details for reasons of space; however, we remark that we don't have a problem with a book in a thought experiment being larger than the known universe.)

Levesque then gives an alternate argument. He suggests that it would be possible to create another kind of instruction manual, one that defines a correct algorithm for addition. We agree that such a book would be easy to create and would be of a modest size, and that a reasonably intelligent human would be able to follow the instructions.

However, he contends, someone using such a book "actually *learns* addition, and not merely a simulation of addition that happens to produce the right external behavior." (Our emphasis.)

Because we argue that the real problem is that Searle's description included a homunculus in the form of an instruction book, we believe that Levesque's rejoinder misses the point. His first Summation Room might be too big to build in our present universe, but given a somewhat bigger one, would produce the desired behavior without necessarily understanding the addition. Where we disagree is with the claim, in his second example, that someone following his algorithm for addition (manipulating formal marks to produce something that looks like a sum) would necessarily understand addition. We no more believe this than we believe, for example, that an electronic calculator *understands* addition. (McCarthy [10] talks of thermostats having beliefs, albeit with a limited range of ideas, but in a way very different from the present discussion.)

Thus, although Levesque's observation that there is no such book is an astonishing insight, we believe that the problem has a different character than Levesque's argument – which makes no mention of a homunculus – would suggest.

2.3 Mechanical reason, revisited

As mentioned earlier, the first order predicate calculus was believed to be a resolution of the paradox of mechanical reason. 'Snow is white' is a syntactic conclusion of a mechanical reasoner if and only if "Snow is white" in the real world described by the semantic interpretation of the symbols. Physical symbol systems abounded during different periods of AI's evolution – scripts, frames, expert systems, rule-based systems – but the soundness and completeness of the first order predicate calculus gave logicians (sometimes referred to as the "logic mafia") a firm foothold in the AI mainstream.

We remark at this juncture that the soundness of and completeness of logic as an answer to the paradox of mechanical reason also introduces a homunculus into the picture. The homunculus enters when we claim there is a perfect, or even a near perfect, assignation of semantics, specifically the meaning of symbols in the real world, and the relationships among them.

Just as there can be no such instruction book in Searle's room, there can be no such map in formal logic, at least at least where pragmatic or even mundane real-world domains are concerned.

Certainly, it is possible to make *certain* maps, some of which are impressive. We can assign semantics from formal marks to concepts such as numbers and objects and sets, and slowly, but surely, build up all of arithmetic, probability theory, geometry, the real numbers, physics, gravity, times, and so on. The peculiar reality is that it is nonetheless hard to get into the subtleties and nuances that simple natural language utterances can carry, not to mention the interpretations imposed by the contextual conditions in which the language is used.

3 THE TURING TEST

Turing, a forerunner both in theories of computing and artificial intelligence wrote a perspicacious paper in *Mind* [13]. Although the computers of his era were toys by modern standards, Turing foresaw their potential.

To explain the digital computer, a brand-new device, Turing described a formalism that came to be known as a Turing machine, which consisted of a read-write head that moves to the right or the left along a one-way infinite tape, and reads, writes or erases zeros, ones, and blanks. A large literature agrees that this Turing machine is, qualitatively, "the best machine in town." It is a straightforward undergraduate exercise to show that extending the Turing machine by making the tape two-way infinite, or planar, adding multiple tapes, creating a random seek head, or adding nondeterminism, may increase the efficiency of the Turing machine, but these enhancements do not extend *what can be computed*. A variety of computing formalisms have come and gone – Markov Algorithms, RAMS – but all have been proven to be equivalent to Turing machines and also do not extend what can be computed.

To any reader of Turing's paper on this subject, it should be clear that the Turing machine itself has no concept of "addition", although it should also be clear that with a little work, one could write a set of rules for a Turing machine that would simulate addition. This may not be recognized as addition by a layperson, but it would be easy enough to create a mapping between the inputs and outputs that would be sound and complete with respect to addition. (We refer the interested reader who wishes to take the time to investigate how addition is implemented on a Turing Machine to go back to Levesque's discussion of the Summation Room that uses a book that describes the addition algorithm, and consider whether an understanding of the corresponding Turing Machine means one understands addition, or whether the Turing Machine itself does.)

That aside, Turing was interested in the idea of machine intelligence, and proposed *the Imitation Game* as a way of testing intelligence without actually defining intelligence – a brilliant finesse of homunculus problems.

3.1 The Gender Game

Turing's first description [14] of the Imitation Game involves a man, a woman, and an interrogator, who is in another room, and who must determine on the basis of text-based conversation which player is the man and which is the woman.

Some writers [5], call the Gender Game a 'red herring', but others [e.g., 2] see it as key to the argument. We agree with the latter view, because we believe the Gender Game suggests that the Turing Test intends to test a kind of intelligence that goes well beyond what might be produced by mechanical means.

Turing uses words along the lines of *pretence* and *imitation*, but, in light of Levesque's arguments against the test, we offer the word *impersonate* – a human player must convincingly inhabit the gender of another *person to* a judge's satisfaction, and furthermore, this illusion must be sustained for a reasonable length of time. We assume that there is no disagreement that sustaining such a convincing performance would require intelligence – whatever we believe that to be.

3.2 The "Standard" Imitation Game

Turing then asks, "What will happen when a machine takes the part of [the man] …?' *Will the interrogator decide wrongly as often* *when the game is played like this as he does when the game is played between a man and a woman?*"

To put it into our terminology, will it be possible for a machine to *impersonate* a human – that is, convincingly inhabit the being of a person, to a judge's satisfaction, and to sustain the illusion for a reasonable length of time?

Turing foresaw that this was a difficult problem. In the original paper, he stated, "I believe that in about fifty years' time it will be possible to programme computers … to make them play the imitation game so well that an average interrogator will not have more than 70 percent chance of making the right identification after five minutes of questioning". It is less well known that in a 1952 radio broadcast [15], he says, in response to a question from Max Newman, that it will be "at least 100 years" before a machine will, in Newman's words, "stand any chance with no questions barred". This is a significant revision and only underscores his insight into mechanical and human intelligence in an era when these concepts were barely understood.

Turing died only two years later. There are no further academic publications from him on this subject, and there appears to be no other historical record of him commenting, publicly, or privately, about this question again.

However, given the history of AI, and some of the other predictions made during its history, it should be added that Turing's predictions were both remarkably conservative and remarkably prescient. The Turing Test has remained a cornerstone of artificial intelligence.

4 LEVESQUE'S CRITIQUE OF TURING

Around the centenary of Turing's birth, Levesque made a strong case that the Turing Test was out of date. This was reported in the mainstream media at a time when *The Imitation Game*, a fictionalized biography of Turing's later life, was a popular film.

According to Levesque [8,9], the Turing Test "has a serious problem: it relies too much on deception". Levesque uses a good deal of loaded language. To wit: "a computer program passes the test iff it can *fool* an interrogator into thinking she is dealing with a person not a computer." A program "will either have to be evasive … or manufacture some sort of false identity (and be prepared to lie convincingly)." "All other things being equal," says Levesque, "we should much prefer a test that did not depend on chicanery of this sort". "Is intelligence just a bag of tricks?" he asks. And so on.

We detect a whiff of moral disapproval.

The Turing Test undoubtedly involves deception, which we prefer to call *impersonation*, for the same reason it involves communication by text. Turing [15] explains: "The new problem has the advantage of drawing a fairly sharp line between the physical and the intellectual capacities of a man." It's the intellectual that interests us. "We do not wish to penalise the machine for its inability to shine in beauty competitions … The conditions of our game make these disabilities irrelevant".

Similarly, questions that amount to asking whether a computer possesses physical attributes of humanness cannot be answered truthfully. If asked, "which tastes better: dark chocolate or milk chocolate?" a computer must give an answer consistent with an ability to taste. Anything else amounts to replying, "yes" to "are you a machine?"

4.1 The Loebner Competition

Levesque's discussion of this "serious problem" [8,9] focuses on the kind of tactics seen in the Loebner competition, which is sometimes understood to be an initial attempt at conducting the [14] Turing Test. The Loebner chatterbots, he says, "rely heavily on wordplay, jokes, quotations, asides, emotional outbursts, points of order, and so on. Everything, it would appear, except clear and direct answers to questions!"

We don't disagree that the Loebner chatterbots currently do not meet the vision of Turing, nor do we feel they satisfy the visions of serious researchers in Artificial Intelligence. Their tactics simply reflect the fact that, in the current state of the technology, the chatterbots are unable to sustain the illusion of humanity for terribly long. We are not alone: the scientific community has not accepted the winner of any Loebner Competition as having passed the Turing test.

Indeed, if this level of discourse were sufficient to pass the Turing Test, Eliza would have been deemed to have succeeded some decades ago.

To combat this "chicanery", Levesque proposes an alternative to both the Loebner Competition and the Turing Test, that avoids the tomfoolery he attributes to the Loebner. Although the design of his test is carefully thought out, our concern is that it, at the same time, cuts the heart out of the Turing Test.

5 LEVESQUE'S PROPOSED ALTERNATIVE

Levesque's proposed alternative to the Turing Test, the Winograd Schema Challenge (WSC), is a constructive one. The WSC is a highly streamlined and uniform multiple-choice exam that allows for a simple scoring mechanism.

Each question is an anaphoric disambiguation test [8] characterised by the inclusion of a *special* word that, when replaced by an *alternate*, flips the answer. In the example below, the *special* word is italicized and its alternate appears, likewise italicized, in parentheses:

Question: The trophy would not fit in the brown suitcase because it was too *big* (*small*). What was too *big (small)?*
  Answer 0: the trophy
  Answer 1: the suitcase.

A WSC would consist of many problems similar to this, which could be answered on the equivalent of bubble answer sheets and marked lickety-split by machines. We concede that the examples provided by Levesque and Winograd are cunning, and that they appear to require real-world knowledge and reasoning to answer correctly.

But therein lurk the homunculi, and they are very difficult to spot. We begin by defining what we call the Watson Effect.

5.1 The Watson Effect

Earlier, we added the words "sustain" and "for a reasonable time" to take into account what we call the *Watson* effect. Readers may recall that IBM built a program called Watson, which was designed to play the TV game *Jeopardy*. (For those unfamiliar with this television program, it is a trivia game where contestants are given “answers” as “clues” in some category and they must reply with the question to which the clue is an answer. For example, a “clue” in the category *Shakespeare* might be “He called these a pair of star-crossed lovers”, to which the correct reply is “Who are Romeo and Juliet?”

In the televised competition between Watson and two human Jeopardy champions, Watson’s performance was almost enchanting until it was given, in the category *U.S. Cities*, the clue: "Its largest airport is named for a World War II hero, its second largest for a World War II battle," Watson answered “Toronto”.

To the North American audience, this was a hysterical blooper, as most of those viewers knew Toronto to be one of the largest cities in *Canada*, not the United States. However, to be fair to Watson, there are several explanations for this gaffe. Many people think of America as comprising all of North and South America. Alternately, because the Toronto Blue Jays baseball team is part of the American League, it may have tripped up. Finally, it turns out that 22 aircraft are based at the Eddie Dew Memorial Airpark located near Toronto, Ohio, USA, the second-largest city in Jefferson County. And so on. But we believe that for most people, Watson’s magic ended then, even though Watson went on to significantly defeat both human jeopardy champions. The same is true of voice recognition functions and almost all varieties of auto-correct: there is a period for which they amaze, a point at which they goof, and, for some, a point at which they annoy.

In the case of Watson, this lone error was sufficient to dispel the illusion of true intelligence. Watson certainly didn’t lose the game – it won by a landslide. But in the judgment of the TV-watching jury, Watson had failed.

For this reason, we have added to our rephrasing of the definition of the Turing Test, that the machine must sustain the illusion of human intelligence for a reasonable length of time.

5.2 The WSC Homunculi

We believe there may be more than one homunculus in the WSC as we understand it, but they seem to be chimeric. The most obvious homunculus is in the scoring mechanism, which we will deal with here. Levesque writes that “a random WS test can be constructed, administered, and graded in a fully automated way. An expert judge is not required to interpret the results.”

For one thing, we don’t believe Turing’s test required an expert judge. It required an ordinary person with no particular expertise in computers as a judge, a group of such persons, as a jury.

More critically, Levesque’s “fully automated grading mechanism” merely gives a grade, and cannot really *decide* intelligence the way Turing’s judge or jury would. It is therefore vulnerable to the Watson effect – a single error may ruin the illusion of intelligence, even though the machine scores well in the test.

Levesque himself observes at least two opposing pitfalls in the design of Winograd Schemas. The first are schemas that are too obvious. He gives the following example:

Question: The women stopped taking the pills because they were *pregnant* (*carcinogenic*). Which individuals were *pregnant* (*carcinogenic*)?

Answer 0: The women.
Answer 1: The pills.

Levesque points out that there is a straightforward mechanism in anaphor resolution for sorting this out (selectional restrictions) because, barring far-fetched readings, only women can get pregnant and only pills can be carcinogenic.

The following illustrates the second pitfall:

Question: Frank was *jealous* (*pleased)* when Bill said that he was the winner of the competition. Who was the winner?

Answer 1: Frank
Answer 2: Bill

There are two possible interpretations of this question. If Frank and Bill are rivals, Frank would be jealous if Bill announced that Bill won the competition. If they were friends, it would be a different story. It is situations like this that make it impossible to write the instruction book for the Chinese room, and that make the Turing Test a meaningful challenge. The machine must be aware of all possible contexts, and in the case of the Turing Test, must have some smart way of disambiguating them – asking questions that clarify the context. In the case of the Chinese room, this would have been included in the instruction manual. In the case of the Turing Test, the machine would have to have the presence to ask a little about Frank and Bill to understand better why one might feel one way or another. But from what appears here, there is no clear correct answer.

In another side remark, Levesque proposes that the questions be designed so that an untrained subject (he suggests “your Aunt Edna”) could answer them all. This is a peculiar suggestion that we believe implies the existence of another homunculus in the argument. Is there really a universal Aunt Edna that all competitors will think like?

These examples, like many examples in the good old days of automated commonsense reasoning, look simple enough at first glance. But there is no clear line in the sand that divides the questions into those for which there is a dead easy answer, those which “your Aunt Edna” would find easy to answer, those which are sufficiently ambiguous to admit several plausible answers, and questions based on knowledge too narrow to expect anyone to know the answer to. But who decides which questions form part of the test? Aunt Edna? No matter how well this is done, some questions will, not by fault of the designers, fall into one or more of these categories. And how does the grading mechanism take into account the normal ambiguity of language in a way that measures the *intelligence* of the software and not just a raw score?

No matter how hard we try, questions will sneak into the test that reflect certain biases the “Aunt Ednas” being used to calibrate the test will introduce.

For Levesque’s grading mechanism to measure intelligence and replace the Turing Test, it must be capable of a kind of discernment that itself requires intelligence; to paraphrase Haugeland, we have presupposed the very intelligence we have proposed to test.

It is also possible for questions to be designed poorly. Perhaps, in the case of Watson, the question about American airports was such a poorly designed question. The history of human intelligence testing abounds with outrageous examples of poorly designed questions that, in some instances, pigeonholed unfortunate groups of people as lacking in intelligence and others as preternaturally intelligent. We cannot presuppose that this kind of mistake will not be made again.

This means that the exam along with the grading mechanism have a chimeric relationship. We might throw Aunt Edna into the mix, but we leave this discussion for future work.

This is not to say that the idea of Winograd Schemas are not ingenious or that the idea of a WSC is not excellent, or that nothing valuable is measured by Levesque's fully automated grading system. What is necessary going forward is that we distinguish *benchmarks* from the judgment of intelligence.

6 BENCHMARKS VERSUS INTELLIGENCE

Over the years, various tasks (usually games) have been proposed as benchmarks for measuring machine intelligence: tic-tac-toe, sliding tile puzzles, chess, checkers, Jeopardy, Go, the Turing Test, the Loebner Competition, and now the Winograd Schema Challenge.

As recently as the 1990s, a brute force tic-tac-toe game implemented in Prolog ran out of stack space in the computers of that era. At that time, tic-tac-toe, along with sliding tile puzzles (aka 16-puzzles) formed a considerable part of the introductory AI curriculum.

Of course, those days are long gone. There is probably little need to optimize search algorithms for a tic-tac-toe implementation. These, and also other far more complex games have fallen, one-by-one, to techniques largely based on massive knowledge bases and brute force searches and sophisticated statistical strategies.

The difference between a benchmark and the Turing Test is that the former has a measurable performance in the formal scientific sense, whereas the latter transcends science – the machine must be perceived to "walk among us" in ways that resist quantification.

Where does this leave the WSC? Does it replace the Turing Test, or is it another benchmark?

The critical issue is scoring. If the WSC is to replace the Turing Test, the scoring mechanism must replace the human judge and decide which participants are playing intelligently, and which are not. If it just provides a score, it is a benchmark.

Lastly, we remark that the Loebner competition is neither – it is a competition. The best effort wins the day, but doesn't necessarily pass the test.

7 HUMAN VALUES AND JUDGING INTELLIGENCE

In one variation of the Imitation Game, Turing proposed judgment by a jury. This is apt because it links the idea of human values in artificial intelligence to the idea of human values in justice.

A Google search using the query 'justice vs law' yields some seventy million hits. For the present discussion, it seems reasonable to say that justice is a philosophical and/or moral concept with no agreed-upon universal definition, whereas the law is a set of rules (presumably) written by in-house government lawyers, and voted (agreed) upon by legislative bodies. Despite all efforts, significant disagreements arise about the interpretation of legislation amongst ordinary people, their lawyers, and many levels of courts.

As well, human values might change over time after the passage of a law. How this happens stretches our expertise, but it is interesting to consider the role juries play in this process.

The jury system is an ancient human innovation [1]. Some scholars date the origin of the modern jury to 1066, but some writers suggest that modern trial by jury would be "unnecessary and burdensome in a primitive state of society, when the family or clan was the social/political unit~~s~~, and laws were few and readily understood."

Interestingly, there are at least two explanations of why juries exist [1]. Some say that juries of peers were introduced to temper the decisions of judges, in the same way the introduction of the House of Commons tempered decisions of the House of Lords.

The other theory, and the one that Burns argues should be *central*, might be characterized as saying that the idea of justice ultimately resides in the minds of humans. This gives juries the right to make decisions that are completely novel, and it also gives juries the right to make decisions that shift as community standards shift.

The jury system, like the Turing Test, has been subject to a variety of criticisms, many of them devastating. Yet in spite of our awareness of imperfections and abuses, most observers would not want to replace something of such value unless and until its fatal defects and the superiority of the proposed alternative were convincingly demonstrated.

Similarly, we view the evaluation of intelligence much as we view the evaluation of justice. Both concepts are difficult to define formally – because both transcend ordinary metrics – but are understood and judged by humans in a way that depends on many factors, including context.

Thus, a philosophical *phase transition* might be said to occur when we move from benchmarks to human judgment. Benchmarks in artificial intelligence, such as winning a tournament, are by definition easy to define and to measure, but a successful outcome does not guarantee we have achieved intelligence. Similarly, no matter how long the law is debated and how carefully its language is constructed, there is no guarantee, for all worlds and times, that we have achieved justice.

This brings us back to our earlier statement, where we stated that the Turing test transcends science. Let us be clear that we do not intend to enter the realm of the supernatural when we say this; it is only that in the Knowledge Representation community, it is appropriate to think of science in terms of the formal logical frameworks articulated, for example, by Kyburg [3]. But consensus on what science is has not been achieved there either – there are also frameworks going back to Popper and Quine, and many variations since.

8 CONCLUSIONS

Levesque has given a strong argument regarding perceived weaknesses within the Turing test; in particular, he objects to the use of "deception" (impersonation) and the free form of the test, which allows contestants to dodge meaningful challenges by changing the topic, or pretending not to understand. He provides a constructive alternative, the Winograd Schema Challenge, which forces the computer to give a direct answer. Moreover, once constructed, this eliminates the need for human judgment.

We counter by saying that this is equivalent to saying there is a computer program that can decide whether an entity is intelligent or not: a regression or homunculus argument. Alternately, it replaces the Turing test, wherein a computer and human “share thoughts” in the judgment of another human, with another benchmark/milestone to pass.

This research was supported by funding from the University of Saskatchewan.

REFERENCES

[1] R.P. Burns, The History and Theory of the American Jury. *California Law Review* 83(6):1477-1494, (1995).

[2] B.J. Copeland, ed., *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma*, Clarendon Press, Oxford, 2004.

[3] D. C. Dennett, "Darwin's dangerous idea." *The Sciences*, 35.3, 34-40, (1995).

[4] J. Haugeland, *Artificial intelligence: The very idea*, MIT press, 1989.

[5] A. Hodges, *Alan Turing: The Enigma*, Random House, 2012.

[6] H.E. Kyburg, Jr., *Theory and Measurement*, Cambridge University Press (2009)

[7] H.J. Levesque, ‘Is it Enough to get the Behavior Right?’, *Proceedings of IJCAI-09, Pasadena, CA*, Morgan Kaufmann, 1439:1444, (2009).

[8] H.J. Levesque, ‘The Winograd Schema Challenge’, *Logical Formalizations of Commonsense Reasoning, 2011 AAAI Spring Symposium, TR SS-11-06*, (2011).

[9] H.J. Levesque, ‘On our best behavior’, *Artificial Intelligence* 212(1): 27-35, (2014).

[10] J. McCarthy, Ascribing mental qualities to machines. In M. Ringle (Ed.), *Philosophical perspectives in artificial intelligence* (pp. 161–195). Atlantic Highlands, NJ: Humanities Press. (1979).

[11] A. Newell and H. Simon, ‘Computer Science as Empirical Inquiry: Symbols and Search’, *Communications of the ACM **19***(3) 113-126 (1976)

[12] J. Searle, ‘Minds, Brains and Programs’, *Behavioral and Brain Sciences*, 3: 417–57, (1980).

[13] J. Searle, ‘Why Dualism (and Materialism) Fail to Account for Consciousness’, in R. E. Lee, ed., Questioning *Nineteenth Century Assumptions about Knowledge, III: Dualism.* NY: SUNY Press, 2010.

[14] A.M. Turing, ‘Computing machinery and intelligence’, *Mind*, 433-460, (1950).

[15] A.M. Turing, R. Braithwaite, G. Jefferson, and M. Newman, ‘Can Automatic Calculating Machines be said to Think?’, (1952), in B.J. Copeland, ed., *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma,* Clarendon Press, Oxford, 487-506, 2004.