STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720(online)
Volume 3, Issue 2
www.stmjournals.com

On the Power of Oracles in the Context of Hierarchical Intelligence

Mark Burgin
Department of Computer Science, University of California, Los Angeles, USA
405 Hilgard Ave. Los Angeles, CA 90095

Abstract

Artificial intelligence (AI) is aimed at the construction of artificial systems with intelligent behavior. At first, AI researchers tried to elaborate a uniform technological (computational) system that had intelligence. However, psychological and neurophysiological studies demonstrated heterogeneous hierarchical organization of the human psyche and intelligence. That is why the contemporary approach to AI is based on heterogeneous hierarchical architecture of computational systems. In this paper, we study the intelligence hierarchy formalized in the framework of the mathematical theory of Oracles.

Keywords: *Artificial intelligence, heterogeneous architecture, hierarchic organization, Oracle, hierarchic intelligence, informative Oracle, power of Oracle*

****Author for Correspondence*** E-mail: mburgin@math.ucla.edu

Intelligence is not the ability to store information, but to know where to find it.
Albert Einstein

INTRODUCTION

Artificial intelligence (AI) has two distinct meanings:

- First, artificial intelligence is interactive intelligence realized by artificial means such as computers, which are now the most appropriate means for this purpose. It is called the technological AI.
- Second, artificial intelligence is a research area aimed at the creation of AI in the first sense. It is called the analytical AI.

There are three approaches to the development of the technological AI:

- Assuming that natural intelligence is a function of the human brain, researchers try to construct AI replicating intelligence of people with the technological core of AI reproducing the structure of the brain.
- Researchers try to construct AI based on the straightforward growth of the potential of technological mechanisms.
- The development of AI goes on as a combination of the first two approaches.

The first approach in AI has been based on artificial neural networks as models of the brain. The second approach in AI has been based on conventional computers, which at first, had one processor (central processing unit or CPU) and then evolved into homogeneous systems of similar processors; cellular arrays of CPUs.

At the same time, in nature homogeneity coexists with heterogeneity. When researchers started to explore the structure of the brain, they found that having high measure of homogeneity on the level of neurons, it is heterogeneous on the level of the brain architecture containing different components and parts, which are organized hierarchically. In the conventional setting, the brain includes three basic components: the forebrain, midbrain, and hindbrain [1].

Forebrain is the largest division of the brain involved in a wide range of activities that make people human. The forebrain has a developed inner structure including the following subcomponents:

- the cerebrum, which consists of two cerebral hemispheres;
- the diencephalon, which is situated under the cerebrum;
- the thalamus, which is the main relay

center between the medulla and the cerebrum;
- the hypothalamus, which is an important control center for sex drive, pleasure, pain, hunger, thirst, blood pressure, body temperature, and other visceral functions;
- the limbic system, which is directly linked to the experience of emotion.

The midbrain is the smallest basic component and it makes connections with the other two basic components; the forebrain and hindbrain, and alerts the forebrain to incoming sensations.

The hindbrain is involved in sleeping, waking, body movements and the control of vital reflexes such as heart rate, blood pressure, and includes the following subcomponents:
- the pons,
- the medulla,
- the cerebellum.

Independent of neurophysiological studies, various computational and communication devices, such as computers, tablets, cell phones and computer networks such as the Internet, are becoming more and more heterogeneous. Recently, the concept of the heterogeneous system architecture was elaborated and utilized for building more efficient and productive computer systems [2]. These systems use more than one kind of processors to achieve higher performance not just by adding cores, but also by incorporating specialized processing devices (typically, CPUs and GPUs) usually on the same silicon die to handle particular tasks. In addition to their 3D graphics rendering capabilities, GPUs can also perform mathematically intensive computations on very large data sets, while CPUs usually run the operating system and perform traditional computational tasks in this heterogeneous processor architecture.

All these finding and developments demonstrate necessity of building artificial intelligence as a heterogeneous hierarchical system, in which computational Oracles play an important role [3–5]. In the context of system theory, the concept of Oracle reflects interaction between systems with less information and systems with more indispensable information where the latter play the role of an Oracle. In such a way, Oracles appear in the hierarchical artificial intelligence where the higher levels function as Oracles for the lower levels. To efficiently organize interactions between levels and achieve better performance of the whole system, it is important to know properties of Oracles and characteristics of their interaction with subordinate systems.

Computational Oracles originated in computer science in the form of Oracle Turing machines. They have been also used in other mathematical models of automata and computations such as inductive Turing machines with Oracles, limit Turing machines with Oracles and evolutionary Turing machines with Oracles [6, 7].

In the process of the development of computer science, researchers achieved a new understanding of computational Oracles extending and enhancing this concept. For instance, Burgin explained that a supercomputer can play the role of an Oracle for an average computer due to the fact that the computing power of the supercomputer can be much higher than that of the average computer [6]. A database can also play the role of an Oracle for an individual or computer, especially, when this database contains data that come from a random source and thus, are incomputable [6, 8].

Independently, computational Oracles have been efficiently used in computational mathematics, where they provided efficient computational schemas for theory and applications [9–11]. Note that Oracles from computational mathematics are essentially different from Oracles in Turing machines.

In addition to abstract Oracles in computer science and software (algorithmic) Oracles in computational mathematics, hardware Oracles have been studied, constructed and used in computer technology. For instance, the Oracle based computing paradigm called DIME network architecture that has been successfully used to implement self-managing distributed systems [12–14]. A DIME network is represented by a grid automaton with such

nodes as DIME units, servers, routers, etc. [6]. Each DIME unit is modeled by a system of basic automata A_i with an Oracle O. The automata A_i model the DIME basic processors P_i, while the Oracle O models the DIME agent DA [12]. The Oracle O in a DIME unit contains information about the intent of the algorithm (along with the context, constraints, communications and control of the algorithm) the basic automata A_i are executing under the control of O and has the visibility of available resources and the needs of the automata A_i as they execute their tasks. In addition, the Oracle O also has the knowledge about alternate courses of action available to facilitate the evolution of the computation to achieve its intent.

Here we study Oracles using methods and constructions of the mathematical theory of Oracles with orientation on the heterogeneous hierarchical artificial intelligence (HHAI). The heterogeneous hierarchical architecture (HHA) of artificial intelligence presupposes a broad spectrum of components on the top level and high homogeneity on the level of elementary elements. In the computational schema based on HHA, some components play the role of Oracles for other components providing a functional hierarchy with respect to the capabilities of the components. Oracles allow formation of efficient hierarchical structures in HHAI.

FUNCTION ORACLES AND THEIR POWER

In the mathematical theory of Oracles, an Oracle M is a system with definite properties with respect to another system T, which can be a human being or an artificial information processing system such as a computer, cell phone, smart phone or computer network.

In this context, it is possible to discern various types and sorts of Oracles. We organize these types and sorts in several classifications starting with the functional typology. Traditionally, it is assumed that the function of Oracles has been providing necessary information to other systems. However, here we treat Oracles as systems with more information or with better information with respect to another system. Such an advantage allows Oracles to perform various functions in the whole system. This brings us to the following classification.

Functional Classification of Oracles

1. An informative Oracle for a system T is a system M that provides some necessary information for T, while T does not have this information.
2. A performing Oracle for a system T is a system M operation of which improves functioning of T.
3. A service Oracle for a system T is a system M that provides services for T that cannot be done by T itself.

The functional classification of Oracles describes functions or roles of Oracles. Here we will study informative Oracles considering different classes of these Oracles.
There are three goal-oriented classes of informative Oracles:

1. A function Oracle M is a system that can provide information about the values of some functions.
2. A property Oracle M is a system that can provide information about properties of some system R.
3. A relation Oracle M for T is a system that can provide information about relations in some system R.

Definition 1
If an Oracle M can provide information about the values of some function f, then f is called an M-function.
An important kind of relation Oracles is a set Oracle.

Definition 2
A set Oracle is a relation Oracle that can provide information about membership in some set X.

Usually, function Oracles and set Oracles are used in computations and complexity theory providing the base for relative computations and hierarchies of degrees [15–18, 8].

Proposition 1
It is possible to reduce any set Oracle to a function Oracle.

Indeed, if a set Oracle *M* provides information about membership in a set *X*, then the function Oracle that provides information about the membership function of the set *X* is performing the same action as *M*.

As any function is a special type of a binary relation, we have the following result.

Proposition 2
It is possible to reduce any function Oracle to a relation Oracle.

In the theory of abstract properties, it is demonstrated that it is possible to represent any relation by an abstract property [6]. It gives us the following result.

Proposition 3
It is possible to reduce any relation Oracle to a property Oracle.

At the same time, it is possible to treat any property as function and thus, as a binary relation. It gives us the following result.

Proposition 4
It is possible to reduce any property Oracle to a relation Oracle.

There are three action types of informative Oracles:
1. A passive Oracle M has the form of stored information and another system has to extract this information.
2. A reactive Oracle M provides information by request.
3. A proactive Oracle M provides information by itself.

This gives us two groups of Oracles: passive Oracles and active Oracles.

The Oracle in an Oracle Turing machine or in an advice-taking machine, is a passive Oracle [19], the Oracle in an Oracle database is a reactive Oracle and the Oracle in DIME architecture, is a proactive Oracle [12].

There are also three substantial classes of informative Oracles:
1. A stored Oracle *M* is storage of information that the Oracle can provide.
2. A computing Oracle *M* is a system that computes information that the Oracle can provide.
3. A transmitting Oracle *M* for *T* is a system that transmits information from some source *P*.

Oracles from all three classes can be in the form of an algorithm, a program for an automaton or an automaton. For instance, when a stored Oracle is in the form of an algorithm, this algorithm describes how to access (or to get) the stored information. When a stored Oracle is in the form of an automaton, this automaton can access and get the stored information. When a stored Oracle is in the form of a program for an automaton, this program describes how the automaton can access and get the stored information.

An important kind of stored Oracles is a tape Oracle.

Definition 3
A tape Oracle is a stored Oracle, in which information is stored (written) in a tape.
Tape Oracles are very popular in theoretical computer science and specifically, in the theory of algorithms, automata and computation. In his paper, Turing introduced the idea of Oracle into theoretical computer science making an obscure remark about o-machines, which are now called Oracle Turing machines [20].

Let us suppose we are supplied with some unspecified means of solving number-theoretic problems; a kind of Oracle as it were.. .. this Oracle. .. cannot be a machine. With the help of the Oracle we could form a new kind of machine (call them o-machines), having as one of its fundamental processes that of solving a given number-theoretic problem.

Later Post formalized, considerably expanded and developed the concept of an Oracle Turing machine [21, 16]. In this setting, an Oracle is a system, e.g., a device, black box, person, whatsoever, that contains knowledge about the values of some, incomputable, function $f(n)$. If a Turing machine *T* has an Oracle for such a function $f(n)$, than *T* has an operation of supplying the Oracle with an arbitrary number n and receiving from the Oracle the value $f(n)$. In such a way, the

Turing machine *T* can compute an incomputable function. Consequently, the Turing Oracle is a function Oracle.

Computer scientists have used Oracle Turing machines for building a sophisticated theory of relative algorithms and computation [17]. In this theory, levels of incomputability are constructed and problems are classified according to these levels [18]. Another area of research in the theory of relative computations studies algorithmic reducibility of one set to another constructing computationally equivalent classes of sets.

Soare even argues that actually relative computations are in the center and form the main content of theoretical computer science [8]. He argues that the notion of an oracle machine and relative computability is the single most important in the subject due to the following reasons:

1. All conventional Turing machines can be easily simulated by Oracle Turing machines and the latter is scarcely more complicated to explain.
2. Most of the objects considered in computability theory and applications to algebra, model theory, geometry, analysis, complexity and other fields are incomputable not computable and relative computability unifies and explains them.
3. Many if not most computing processes in the real world are online or interactive processes, better modeled by Oracle Turing machines than by conventional Turing machines.

It is interesting that there are other models of computation, which are more constructive but have the same abilities as Turing machines with arbitrary Oracles. For instance, general inductive Turing machines have the same computing power as Oracle Turing machines, that is, they can compute any function for finite words and decide any formal language with a finite alphabet [6]. Another model, which has the computing power of Oracle Turing machines, is an advice-taking Turing machine [19, 22].

An advice-taking Turing machine is a Turing machine enhanced with a tape where the values of the advice function are stored $f(x)$ and with the possibility to access the tape with the advice in constant time and read from it the value of its advice function $f(x)$ also in constant time. Consequently, the Oracle in an advice-taking Turing machine is a tape Oracle. The construction of an advice-taking Turing machine coincides with the contemporary formalization of an Oracle Turing machine [23, 8]. Advice-taking Turing machines are important in complexity theory because definitions and results are often based on special Turing machines that can determine information from the Oracle in constant time.
The theory of relative computability, Oracle Turing machines and advice-taking Turing machines provide a good mathematical framework for many cases of database or online computing just as conventional Turing machines provide one for offline computing processes, such as batch processing [8]. This reflects the situation that Oracle Turing machines and advice-taking Turing machines acquire higher computing power from outer sources, such as an Oracle, although in the theoretical model these sources may be attached to the abstract automaton.

It is possible to discern function Oracles by the number of functions information about which these Oracles provide. There are univalent function Oracles, which can provide (information about) the values of a single function, and polyvalent function Oracles, which can provide (information about) the values of several functions. For a univalent function Oracle *M*, it is possible to treat the value $M(a)$ as the value $f(a)$ of the *M*-function *f*.

Definition 4
If O is a system of function Oracles, then a function Oracle *M* is called a comprising Oracle for the system O if for any Oracle *P* from O, any *P*-function is also an *M*-function.
When a function Oracle *M* is comprising for a system of function Oracles E, we denote this by $M \supseteq E$ or $E \sqsubseteq M$.

It is possible to consider the class of all deterministic Turing machines T as the system of function Oracles O. In this case, the *T*-function of a Turing machine *T* is the function

computed (or accepted by *T*. Then any universal Turing machine *U* is comprising for the class T.

Let us consider two systems of function Oracles O and E.

Proposition 5
If $O \subseteq E$, then any function Oracle *M* that is comprising for E is also comprising for O.

Indeed, a function Oracle *M* that is comprising for E can provide (information about) the values of any *A*-function for any Oracle *A* from O.

Definition 4 also implies the following result.

Proposition 6
If a function Oracle *M* is comprising for E and O, then it is also comprising for $O \cup E$.

Propositions 5 and 6 imply the following result.

Theorem 1:
The set of all systems of function Oracles for which an Oracle *M* comprises form a set-theoretical ideal in the class of all systems of function Oracles [24].

Let us consider a system of function Oracles $O=\{O_i;\ i \in I\}$ and a system of function Oracles E.

Proposition 7
If for each system O_i, there is an Oracle in E that is comprising for O_i, then any function Oracle *M* that is comprising for E is also comprising for O.

Indeed, if *f* is a *P*-function of an Oracle *P* from O, then *P* belongs to some system O_i, and there is an Oracle *K* in E that is comprising for O_i, i.e., *f* is also a *K*-function. As the Oracle *M* that is comprising for E, *f* is as well an *M*-function. Consequently, *M* is comprising for O.

Definition 5
A function Oracle *M* is more powerful than a function Oracle *T* if it is comprising for *T*.

When a function Oracle *M* is more powerful than a function Oracle *T*, we denote this by $M \succcurlyeq T$ or $T \preccurlyeq M$.

Definitions 4 and 5 imply the following result.

Lemma 1
A function Oracle *M* is more powerful than a function Oracle *T* if any *T*-function is also an *M*-function.

The relation to be more powerful is transitive as Proposition 7 implies the following result.

Proposition 8
If a function Oracle *M* is more powerful than a function Oracle *T* and the function Oracle *T* is more powerful than a function Oracle *P*, then the function Oracle *M* is more powerful than the function Oracle *P*.

Corollary 1
The relation to be more powerful is a preorder in the classes of function Oracles.

It is also possible to consider power of systems of function Oracles.

Definition 6
A system of function Oracles E is more powerful than a system of function Oracles O if for any Oracle *K* in O, there is a more powerful Oracle *T* in E.

When a system of function Oracles O is more powerful than a system of function Oracles E, we denote this by $O \succcurlyeq E$ or $E \preccurlyeq O$.

Lemma 2
For any systems of function Oracles E and O, $O \sqsupseteq E$ implies $O \succcurlyeq E$.

The relation to be more powerful for systems of function Oracles is transitive as Proposition 8 implies the following result.

Proposition 9
If a system of function Oracles E is more powerful than a system of function Oracles O and the system of function Oracles O is more powerful than a system of function Oracles H, then the system of function Oracles E is more powerful than the system of function Oracles H.

Corollary 2
The relation to be more powerful is a preorder in the class of all systems of function Oracles.
Definitions 6 implies the following result.

Proposition 10
If a system of function Oracles E is more powerful than a system of function Oracles O and a system of function Oracles H, then the system of function Oracles E is more powerful than the system of function Oracles O∪H.

It is possible to extend this result to any set of systems of function Oracles.

Proposition 11
If a system of function Oracles E is more powerful than a system of function Oracles O and H⊆E, then the system of function Oracles E is more powerful than the system of function Oracles H.
The power of function Oracles depends on the functions about which they provide information. This brings us to the following definition.

Definition 7
A function Oracle *M* is instructive with respect to a set F of functions if it can provide (information about) the values of any function from F.

When a function Oracle *M* is instructive with respect to a set F, we denote this by:

$$M \sqsupseteq F \text{ or } F \sqsubseteq M$$

For instance, any total, i.e., everywhere defined, algorithm is instructive with respect to the function it computes.

Note that being instructive with respect to a function *f*, means that the Oracle can provide information not only about the values where *f* is defined but also can inform when *f* is not defined.

Let us consider two systems of functions F and G.

Proposition 12
If F⊆G, then any function Oracle *M* that is instructive with respect to G is also instructive with respect to F.

Proposition 13
If a function Oracle *M* is instructive with respect to F and G, then it is also instructive with respect to the union F∪G.

Propositions 12 and 13 imply the following result.

Theorem 2
The set of all sets of functions for which an Oracle *M* is instructive form a set-theoretical ideal in the class of all sets of functions.

Let us consider two functions f: $X \to X$ and g: $X \to X$ where X is a set.

Proposition 14
If a function Oracle *M* is instructive with respect to *f* and *g*, then it is also instructive with respect to their sequential composition $f \circ g$.

Indeed, a function Oracle *M*, which is instructive with respect to *f* and *g*, can provide information about the value $(f \circ g)(a)$ giving, at first, the value $g(a)$ because *M* is instructive with respect to the function *g*, and then the value $f\ (g(a))$ because *M* is instructive with respect to the function *f*.

Note that obtaining information from the Oracle *M* about the sequential composition $f \circ g$ can require more than twice as many enquiries as it is done for each function *f* and *g*.

Definition 8
A function Oracle *M* is weakly instructive with respect to a set F of functions if it can provide (information about) the values of any function from F when this function is defined for the corresponding argument.

When a function Oracle *M* is weakly instructive with respect to a set F, we denote this by $M \ni F$ or $F \Subset M$.

For instance, any algorithm is weakly instructive with respect to the function it computes.

We see that any function Oracle *M*, which is instructive with respect to a set F of functions, is weakly instructive with respect to F, i.e., $M \sqsupseteq F$ implies $M \ni F$.

There is a natural relation between comprising and instructive Oracles.

Lemma 3
A function Oracle M is comprising for a system O of function Oracles if and only if it is instructive for any function, for which some Oracle from O is instructive, and is weakly instructive for any function, for which some Oracle from O is weakly instructive.

Indeed, being able to provide the same information as Oracles from O means the Oracle M is instructive or correspondingly, weakly instructive for the same functions as Oracles from O.

Let us consider two systems of functions F and G.

Proposition 15
If $F \subseteq G$, then any function Oracle M that is weakly instructive with respect to G is also weakly instructive with respect to F.

Definitions imply the following result.

Proposition 16
If a function Oracle M is weakly instructive with respect to F and G, then it is also weakly instructive with respect to $F \cup G$.

Propositions 15 and 16 imply the following result.

Theorem 3
The set of all sets of functions for which an Oracle M that is weakly instructive form a set-theoretical ideal in the class of all sets of functions.

Let us consider two functions f: $X \to X$ and g: $X \to X$ where X is some set.

Proposition 17
If a function Oracle M is weakly instructive with respect to f and g, then it is also weakly instructive with respect to their sequential composition $f \circ g$.

Indeed, if a function Oracle M is weakly instructive with respect to functions f and g, then it can at first, provide information about the value of the function g at a point a and then provide information about the value of the function f at a point $g(a)$ giving in such a way information about the value of the function $f \circ g$ at a point a.

To study populations (systems) of Oracles, we introduce definite axioms considering Oracles M with M-functions of the form M: $\boldsymbol{N \to N}$. The first problem, which we are going to treat, asks when given a system of Oracles O, it is possible to find in O an Oracle comprising the whole system O.

Axiom D1
For any systems of univalent function Oracles E and O, which provide information about functions on whole numbers W, if $E \subseteq O$ and E is enumerable, i.e., a projection en: $W \to E$ is determined, then there is an Oracle A from O such that for any $n \in N$, $A(n) \neq [en(n)](n)$.

The set of all Oracles such that each contains values of a general recursive functions of one argument is an example of a system of function Oracles, which satisfies Axiom D1. The set of all total Turing machines TT with one tape and one head is another example of a system of Oracles, which satisfies Axiom D1.

Proposition 18
If Axiom D1 is true for the system of univalent function Oracles O, then O is not enumerable.

Indeed, if we suppose that system O is enumerable i.e., there is a projection en: $W \to O$, then by Axiom D1, there is an Oracle A from O such that for any $n \in N$, $A(n) \neq [en(n)](n)$. However, the A-function cannot be equal to the M-function of any Oracle M from O. This contradiction completes the proof.

Definition 9
A function Oracle M constructively comprises a systems of function Oracles O if given a number n and a name of an Oracle P from O, M can provides the value $P(n)$.

Axiom D1 implies some restrictions on the class O.

Theorem 4
If Axiom D1 is true for a system of function Oracles O, then O does not contain an Oracle that constructively comprises O.

Informally, it means that in of function Oracles that satisfies Axiom D1, there is no

Oracle comprising this system, i.e., an Oracle that can provide information that has any other Oracle in this system. Note that it is possible that such a comprising Oracle exists outside the initial system. For instance, the system of all total Turing machines TT does not have a comprising Oracle in TT but a universal Turing machine is a comprising Oracle for TT. Note that it is possible that there is an Oracle that constructively comprises O but it does not belong to O. For instance, a universal Turing machine is a comprising Oracle for the class of all total Turing machines but it is not total [6].

At the same time, we can ask a simpler question, namely, whether it is possible to find an Oracle that can discern one value from all other values provided by Oracles from the same system. Let us assume that function Oracles from O provide information about functions of the form f: $Z \to X$.

Definition 10
(a) An Oracle M is discerning in a system of function Oracles O for an element a from X if given a name of a P-function f of an Oracle P from O and an element b from Z, the Oracle M can provide information whether $f(b)=a$ or not.

(b) An Oracle M is discerning for a system of function Oracles O if given a name of a P-function f of an Oracle P from O and two elements a from X and b from Z (e.g., whole numbers n and m when X is the set of whole numbers W), the Oracle M can provide information whether $f(b)=a$ or not.

Axiom D2
For any two Oracles M and K from the system of function Oracles O, there is an Oracle A from O that comprises M and K.

The set of all Oracles such that each contains values of a general recursive functions is an example of a system of function Oracles, which satisfies Axiom D2. The set of all total Turing machines TT is another example of a system of Oracles, which satisfies Axiom D2.

Definition 11
If f: $Z \to X$ is a function and $c_{a,\ b}$: $X \to X$ is a mapping that for two elements a and b from X, maps a into b and b into a being identical on other elements, then the composition $f \circ c_{a,\ b}$ is called a binary transposition $f_{a,\ b}$ of f.

Axiom D3
For any Oracle M from the system of function Oracles O and any binary transposition $f_{a,\ b}$ of f of an M-function, it is possible to find (the name of) an Oracle A from O an A-function of which is equal to $f_{a,\ b.}$

Lemma 4
If Axiom D3 is true for the system of function Oracles O, then any Oracle M that is discerning for an arbitrary element a from X is discerning for any other element from X.

Proof: Let us assume that an Oracle M is discerning in O for an arbitrary element a from X and d is another element from X. Taking an Oracle P from O, we find the name of a P-function f which we want to test. Using Axiom D3, we find the name of the Oracle A from O an A-function of which is equal to $f_{a,\ d}$. Then given a name of an Oracle A from O, the Oracle M can provide information whether $f_{a,\ d}(b)=a$ or not. However, $f_{a,\ d}(b)=a$ if and only if $f(b)=d$. In such a way, the Oracle M provides information whether $f_{a,\ d}(b)=d$ or not. As d is an arbitrary element from X, the Oracle M is discerning for a systems of function Oracles O.

Lemma is proved.

Corollary 4
If Axiom D3 is true for the system of function Oracles O, then any Oracle M that is discerning for an arbitrary element a from X is discerning for the system O.

Lemma 5
If Axiom D2 is true for the system of function Oracles O and O has an Oracle for the function $c_{a,\ b}$, then Axiom D3 is true for the system O.

Proof: Let us assume that f is an M-function of an Oracle M and O has an Oracle for the function $c_{a,\ b}$. Then by Axiom D2, there is an Oracle A for which both f and $c_{a,\ b}$ are A-function. By Proposition 14, $f \circ c_{a,\ b}$ is also an A-function. Thus, Axiom D3 is true for the system O.

Lemma is proved.

Let us assume that Z=W, elements 0 and 1 belong to X and informing that $f(b)=a$, the discerning Oracle M_a gives 1 as its output and 0 otherwise.

Axiom D4
If an Oracle M from the system of function Oracles O can provide information about values of functions f_n where n=1, 2, 3, … given the number of the function and its argument, then there is an Oracle A from O an A-function g of which is defined as $g(n)=f_n(n)$.

Axiom D5
The system F_O of functions, for which Oracles from O are instructive, is enumerable, i.e., there is a projection en: $W\rightarrow$E.

Theorem 5
If Axioms D3, D4 and D5 are true for a system of function Oracles O, then O does not have a discerning Oracle for any element a from X.

Proof: Let us assume that a system of function Oracles O has a discerning Oracle M_0 for the element 1. By Axiom D5, we have $F_O=\{f_n$ where n=1, 2, 3, … } and given a whole number n, the Oracle M can provide information whether $f_n(n)=0$ or not. By Axiom D4, there is an Oracle A from O an A-function g of which is defined as $g(n)=f_n(n)$. As g belongs to F_O, we have $g=f_m$ for some whole number m.

Let us assume that $f_m(m)=0$. Then given the input (m, m), the Oracle M_0 gives the output 1. However, in this case, $g(m)=f_m(m)$ has the value 1 by its definition. This contradiction shows that $f_m(m)\neq 0$. In this case, the Oracle M_0 gives the output 0 and consequently, $g(m)=f_m(m)$ has the value 0 by its definition. This contradiction shows that it is also impossible that $f_m(m)\neq 0$.

As either $f_m(m)=0$ or $f_m(m)\neq 0$ is true, such an Oracle M_0 does not exist. By Lemma 4, if an Oracle M is discerning for an element a from X, then it is discerning for the element 1 from X. Thus, the system O does not have a discerning Oracle for any element a from X.

Theorem is proved.

Corollary 3
If Axioms D3, D4 and D5 are true for the system of function Oracles O, then O does not have a discerning Oracle.

Corollary 3
If Axioms D2, D4 and D5 are true for the system of function Oracles O and O has an Oracle for the function $c_{a,\,b}$, then O does not have a discerning Oracle.

Note that it is possible that there is a discerning Oracle for O but it does not belong to O. For instance, a relevant inductive Turing machine can be a discerning Oracle for the class of all Turing machines [6].

It would be also interesting to find an Oracle that can discern when an arbitrary Oracle from O can provide information about the values of its function and when it cannot. We call such an Oracle a definability discerning Oracle for O. To explore this possibility, we introduce one more axiom.

Axiom D6
If f is an M-function of an Oracle M from O and h is a function that is undefined when f takes the value 1 and coincides with f in all other cases, then g is an A-function of some Oracle A from O.

Theorem 6
If Axioms D2 and D6 are true for a system of function Oracles O, then O does not have a definability discerning Oracle.

Proof: Let us assume that a system of function Oracles O has a definability discerning Oracle M and f is the M-function of M. By Axiom D6, there is an Oracle A from O, the A-function g of which is undefined when f takes the value 1 and coincides with f in all other cases. By Axiom D2, there is an Oracle A in O for which both f and g are A-function. By Proposition 14, $g\circ f$ is also an A-function.

As A belongs to O, it has some number m. By our assumption the Oracle M defines whether A is defined for the argument m or not, giving 1 when $A(m)$ is defined and giving 0 when $A(m)$ is undefined.

At first, let us assume that $A(m)$ is defined. Then given the input (m, m), the Oracle M

gives the output 1, while the Oracle A becomes undefined. This contradiction shows that $A(m)$ cannot be defined. So, we assume that $A(m)$ is undefined. Then given the input (m, m), the Oracle M gives the output 0, while the Oracle A gives the same output. It means that $A(m)$ is defined. This contradiction shows that $A(m)$ cannot be undefined. As $A(m)$ is either defined or undefined, such an Oracle M_0 does not exist.

Theorem is proved.

Note that there are also classes of function Oracles that have definability discerning Oracles. For instance, the class of all total Turing machines has a definability discerning Oracle because all functions defined by these machines are total and so, the definability discerning Oracle takes only one value 1.

It is also necessary to remark that it is possible that there is a definability discerning Oracle for O but it does not belong to O. For instance, a relevant inductive Turing machine can be a definability discerning Oracle for the class of all Turing machines [6].

REFERENCES

1. DeArmond SJ, Fusco MM, Dewey M. *Structure of the Human Brain*: *A Photographic Atlas*. New York, NY, USA: Oxford University Press; 1989.
2. Hwu W. *Heterogeneous System Architecture, A New Compute Platform Infrastructure*. Morgan Kaufmann; 2015.
3. Myers KL. Strategic Advice for Hierarchical Planners. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers Inc.; 1996; 112–123p.
4. Dietterich TG. Hierarchical Reinforcement Learning with the maxq Value Function Decomposition. *J Artif Intell Res*. 2000; 13: 227–303p.
5. Livingstone D. Coevolution in Hierarchical AI for Strategy Games. In *IEEE Symposium on Computational Intelligence and Games (CIG05)*, Essex, UK, IEEE. 2005; 190–194p.
6. Burgin M. *Super-Recursive Algorithms*. New York: Springer; 2005.
7. Burgin M, Eberbach E. On Foundations of Evolutionary Computation: An Evolutionary Automata Approach. In: Hongwei Mo, editor. *Handbook of Research on Artificial Immune Systems and Natural Computing*: *Applying Complex Adaptive Technologies*. Hershey, Pennsylvania: IGI Global; 2009; 342–360p.
8. Soare RI. Turing Oracle Machines, Online Computing, and Three Displacements in Computability Theory. *Ann Pure Appl Logic*. 2015; 160: 368–399p.
9. Edmonds J. Minimum Partition of a Matroid into Independent Subsets. *J Res Natl Bur Stand.* 1965; 69B: 67–72p.
10. Edmonds J. Matroids and the Greedy Algorithm. *Math Prog*. 1971; 1: 127–136p.
11. Vondrak J. Optimal Approximation for the Submodular Welfare Problem in the Value Oracle Model. *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC'08).* 2008; 67–74p.
12. Burgin M, Mikkilineni R. Semantic Network Organization based on Distributed Intelligent Managed Elements. In *Proceeding of the 6th International Conference on Advances in Future Internet*, Lisbon, Portugal. 2014; 16–20p.
13. Burgin M, Mikkilineni M, Morana G. From Personal Computers to Personal Computing Networks: A New Paradigm for Computation. In *Proceeding of Future Computing 2015, The Seventh International Conference on Future Computational Technologies and Applications*. 2015; 24–30p.
14. Mikkilineni R, Morana G, Burgin M. Oracles in Software Networks: A New Scientific and Technological Approach to Designing Self-Managing Distributed Computing Processes. *Proceedings of the 2015 European Conference on Software Architecture Workshops*, Dubrovnik/Cavtat, Croatia, ACM. 2015; 11:1–11:8p.
15. Basu S. On the Structure of Subrecursive Degrees. *J Comp Syst Sci*. 1970; 4(5): 452–464p.
16. Post EL. Degrees of Recursive Unsolvability, Preliminary Report. *Bull*

Amer Math Soc. 1948; 54: 641–642p.
17. Rogers H. *Theory of Recursive Functions and Effective Computability*. Cambridge Massachusetts: MIT Press; 1987.
18. Shoenfield JR. On Degrees of Unsolvability. *Ann Math*. 1959; 69: 644–653p.
19. Balcazar JL, Diaz J, Gabarro J. *Structural Complexity*. Berlin/Heidelberg/New York: Springer-Verlag; 1988.
20. Turing AM. Systems of Logic Defined by Ordinals. *Proc Lond Math Soc*. *Ser. 2*. 1939; 45: 161–228 p.
21. Post EL. Recursively Enumerable Sets of Positive Integers and Their Decision Problems. *Bull Amer Math Soc*. 1944; 50: 284–316p.
22. Schöning U. Complexity Theory and Interaction. In *The Universal Turing Machine: A Half-Century Survey*, Oxford University Press, Oxford. 1988; 561–580p.
23. Homer S, Selman AL. Relative Computability. In *Computability and Complexity Theory*, Series Texts in Computer Science, Springer. 2011; 145–179p.
24. Kuratowski K, Mostowski A. *Set Theory*. Amsterdam: North Holland P.C.; 1967.
25. Burgin M. *Measuring Power of Algorithms, Computer Programs, and Information Automata*. New York: Nova Science Publishers; 2010.

Cite this Article
Mark Burgin. On the Power of Oracles in the Context of Hierarchical Intelligence. *Journal of Artificial Intelligence Research & Advances.* 2016; 3(2): 6–17p.