Dynamic Logic of Phenomena and Cognition

Boris Kovalerchuk, Leonid Perlovsky

Abstract—**Modeling of complex phenomena such as the mind presents tremendous computational complexity challenges. The neural modeling fields theory (NMF) addresses these challenges in a non-traditional way. The main idea behind success of NMF is matching the levels of uncertainty of the problem/model and the levels of uncertainty of the evaluation criterion used to identify the model. When a model becomes more certain then the evaluation criterion is also adjusted dynamically to match the adjusted model. This process is called dynamic logic (DL) of model construction, which mimics processes of the mind and natural evolution. This paper provides a formal description of Phenomena Dynamic Logic (P-DL) and outlines its extension to the Cognitive Dynamic Logic (C-DL). P-DL is presented with its syntactic, reasoning, and semantic parts. Computational complexity issues that motivate this paper are presented using an example of polynomial models**.

1. Introduction

To provide a formal description of Phenomena Dynamic Logic (P-DL) and its extension to the Cognitive Dynamic Logic (C-DL) we start from background definitions of DL, logic model theory, and their comparisons. Section 2 establishes concepts of uncertainty, generality, and simplicity for models and evaluation criteria. Section 3 defines a partial order of models. Section 4 provides examples of uncertainty and generality of polynomial models. Section 5 formalizes the similarity maximization. Section 6 defines DL parameterization using the theory of monotone Boolean functions. Section 7 defines search process. The core of the paper is section 8 that provides a formal description of Phenomena Dynamic Logic (P-DL) and outlines its extension to the Cognitive Dynamic Logic (C-DL). Section 9 summarizes the paper and discusses future research. The Appendix describes computation complexity issues that motivate this paper.

We start from defining a concept of empirical data relevant to the neural modeling fields (NMF) [8], [9] and its interpretation in logical terms.

Manuscript received December 17, 2007.

Boris Kovalerchuk is with the Air Force Research Laboratory, Sensors Directorate, Hanscom AFB on leave from the Dept. of Computer Science, Central Washington University, Ellensburg, WA 98926-7520, USA (phone: 509 963-1438; fax: 509 963-1449; e-mail: borisk@cwu.edu).

Leonid Perlovsky is with the Harvard University and the Air Force Research Laboratory, Sensors Directorate, Hanscom AFB (leonid@seas.harvard.edu).

Empirical data, E in NMF is any data to identify a model.

In logical terms we define it as a pair **E**=<A, Ω>, where A is a set of objects, Ω={P_i } is a signature, that is a set of predicates P_i of arity n_i, e.g., $P_1(x,y)$ with $n_i = 2$ can mean that length of x is no less than the length of y, $l(x) \geq l(y)$.

Definition. A pair <A, Ω> is called an ***empirical system*** [6].

Definition. A pair <A, Ω> is known in logic as **a *model* (of the system of axioms T)** [7].

Tarski proposed the name 'model theory' in 1954. A variety of other names are also used that include a *relational system* [6], and a *protocol of the experiment*. To distinguish this model from a model in NMF we will also call this model a **logic model (L-model)** or first-order model [10]. Logic models and algebraic methodology found many applications inside of mathematics and other fields such as system software.

The next important concept in NMF is a concept of **a priori model** (of reality), M. In logic formalization, it can be matched with a **system of axioms** T.

Definition. A **system of axioms** T is a set of closed first order logic (FOL) formulas (sentences) in the signature, e.g., $\forall x_i \exists x_j P_1(x_i,x_j)$.

These two concepts are treated very differently in NMF and logic. Logic is going from a *very formal* (syntactical) axiomatic system T to something more real called a model $\mathbf{A_T}$ =<A_T, Ω> of that formal system T. NMF is going in the opposite way -- from *very informal* reality to more formal models. As a result, the concepts of a model are quite different in two theories. Empirical data in NMF is a model **E**=<A, Ω> in logic, if we interpret empirical data as an empirical system E [6]. On the other hand, NMF model is not a model in logic it leans rather to a set of axioms about the class of the logic models. This type of difference was well described in [10]: "To model a *phenomenon* is to construct a formal theory that describes and explains it. In a closely related sense, you *model* a system or structure that you plan to build, by writing a description of it. These are very different senses of 'model' from that in model theory: the '**model' of the phenomenon** or the system is not a structure but a *theory*, often in a formal language." For short, we call a model of phenomenon a **P-model** in contrast with a logic model denoted as an **L-model**.

The next NMF concept is a **similarity (**or **correspondence) measure** L(M,E) between empirical data E and a priory model M that is assigned individually to each specific problem and E: L: {(M,E)} → R, where R is a set or real numbers. In logic the closest to it is a statement that M =<A,Ω> is a model of the system of the axioms T.

Definition. Pair E =<A,Ω> is a **L-model of the system of the axioms** T if every formula from T is true on E.

Definition. **Boolean similarity measure** B(T,E) is defined to be equal to 1, B(T,E)=1, If M is an L-model of T, else B(T,E)=0.

II. SEMANTIC CONCEPTS OF UNCERTAINTY, GENERALITY AND SIMPLICITY

A. Uncertainty, generality and simplicity relations between P-models

Below we introduce the concepts of uncertainty, generality, and simplicity relations. These concepts can be applied to both logic and NMF models.

An **uncertainty relation between P-models** is denoted as $\geq_{Mu}$, relation $M_i \geq_{Mu} M_j$ is read: "Model M_i is equal in uncertainty or *more uncertain* than model M_j" or "Model M_j is no less certain than model M_j". This relation is a partial order.

A **generality relation between P-models** is denoted as $\geq_{Mg}$ and relation $M_i \geq_{Mg} M_j$ is read: "Model M_j is a *specialization* of the measure M_i" or "Model M_i is a *generalization* of the measure M_j". This relation also is a partial order.

A **simplicity relation between P-model** is denoted as $\geq_{Ms}$ and relation $M_i \geq_{Ms} M_j$ is read: "Model M_i is equal in simplicity of *simpler* than Model M_j". This relation also is a partial order.

For NMF models that are represented as a system of axioms the generality relation can be defined as follows.

Definition. $T_i \geq_{gen} T_j$ if and only if $T_i \subset T_j$, i.e., system of axioms T_i is equal to or more general than the system of axioms T_j if and only if T_i contains less axioms than T_j.

B. Uncertainty, generality and simplicity relations between similarity measures

Concepts introduced below can be applied to both logic and NMF similarity measures.

An **uncertainty relation** between similarity measures is denoted as $\geq_{Lu}$ and relation $L_i \geq_{Lu} L_j$ is read: "Measure L_i is equal to in uncertainty or more uncertain than measure L_j". This is a partial order relation.

A **generality relation between similarity measures** is denoted as $\geq_{Lg}$ and relation $L_i \geq_{Lg} L_j$ is read: "Measure L_j is a *specialization* of measure L_i and measure L_i is a *generalization* of the measure L_j" This relation also is a partial order.

A **simplicity relation between similarity measures** is denoted as $\geq_{Ls}$ and relation $L_i \geq_{Ls} L_j$ is read: "Measure L_j is equal in simplicity or *simpler* than measure L_i". This relation also is a partial order.

Definition. Mapping F between a **set of P-models** {M} and a set of similarity measures {L}

$$F: \{M\} \rightarrow \{L\}$$

is called a **match mapping** if F preserves uncertainty, generality and simplicity relations between models and measures in the form of *homomorphism* from a relational system $< \{M\}, \geq_{Mg}, \geq_{Mu} >$ to a relational system $< \{L\}, \geq_{Lg}, \geq_{Lu} >$, i.e.,

$$\forall M_a, M_b \ (M_a \geq_{Mg} M_b \Rightarrow F(M_a) \geq_{Lg} F(M_b)),$$

$$\forall M_a, M_b \ (M_a \geq_{Mu} M_b \Rightarrow F(M_a) \geq_{Lu} F(M_b)).$$

III. PARTIAL ORDER OF P-MODELS

Two different models can be at the same level of uncertainty ($M_1 =_u M_2$), one model can be more uncertain than another one ($M_1 >_u M_2$), or these models can be incomparable for uncertainty.

We may define *model uncertainty* in such way that two different quadratic models M_1: $2x^2 + 3y$ and M_2: $5x+4y^2$ will have the same level of uncertainty $M_1 =_u M_2$. The *number of unknown coefficients* is a possible ways to do this. For M_1 and M_2, these numbers m_1 and m_2 are equal to zero. All coefficients are known and models are certain.

Definition. *NUC measure of polynomial model uncertainty* is defined as the Number of Unknown Coefficients (NUC) in the model.

The generality relation between models M_1 and M_2 can also be defined. For instance, it can be the *highest power n* of the polynomial model. Both models M_1 and M_2 are quadratic with $n_1=n_2=2$ and, thus, have the same generality.

Definition. *HP measure of polynomial model generality* is defined as the *Highest Power n* of the polynomial model.

Alternatively, we may look deeper and notice that M_1 contains x^2 and M_2 contains y^2. We may define the *generality* of a polynomial model as its *highest polynomial variable,* which are x^2 for M_1 and y^2 for M_2. We cannot say that one of them is more general and can call them incomparable in generality.

Definition. *HPV measure of polynomial model generality* is defined as the *Highest Power Variable (HPV)* of the polynomial model.

Consider model M_3: $5x + by^2$. Using NUC measure this model is *more uncertain* than model M_2: $5x+4y^2$, $M_3 >_{Mu} M_2$, because coefficient b in M_3 is not known, that is NUC for M_2 is $n_2=0$ and NUC for M_3 is $n_3=1$ and $n_3>n_2$.

We can also call M_3 *more general* than M_2: $5x+4y^2$, $M_3 >_{Mg} M_2$, because M_2 is a specialization of M_3 with b=4. Similarly model M_4: $ax+cx^2+by^2$ is more general and uncertain than models M_1: $2x^2 + 3y$, M_2: $5x+4y^2$ and M_3: $5x + by^2$, because all coefficients in M_4 are uncertain, but none of the coefficients is uncertain in M_1, M_2 and M_3. In these examples, uncertainty and generality relations are *isomorphic* (produce the same order of models) and it is hard to distinguish them. In the next section, we provide a parameterization mechanism that highlights the difference.

IV. PARAMETERIZATION

Below we parameterize uncertainty and generality of polynomial models. Consider an example of models with increasing levels of uncertainty starting from zero uncertainty at level 0:

Level 0: $3x+4y+5y^2$. All coefficients are known.
Level 1: $ax+4y+5y^2$. One coefficient is unknown.
$ax+9y+5y^2$.
Level 2: $ax+by+5y^2$. Two coefficients are unknown.
$ax+by+7y^2$.

Models $3x+4y+5y^2$, $ax+4y+5y^2$ and $ax+by+7y^2$ form a chain from a more specific and certain model (level 0) to a less specific and certain model (level 2).

In contrast models $3x+4y+5y^2$, $ax+9y+5y^2$ and $ax+by+7y^2$ form an *increasing uncertainty chain* by UNC measure, but they *do not form an increasing generality*. We cannot get $3x+4y+5y^2$ by specializing $ax+9y+5y^2$ and $ax+by+7y^2$, because y coefficients 4 and 9 are different. Similarly, we cannot get $ax+9y+5y^2$ by specializing $ax+by+7y^2$. Thus, we see that uncertainty and generality relations are different.

Another example provides to us the five models at five uncertainty levels:

Uncertainty level n=0: $M_0= 9x^2+3y+7x+10$
Uncertainty level n=1: $M_1= x^2+3y+7x+10$
Uncertainty level n=2: $M_2= ax^2+3y+7x+10$
Uncertainty level n=3: $M_3= ax^2+by+7x+10$
Uncertainty level n=4: $M_4=ax^2+by+cx+d$

Models M_4, M_3, M_2, M_1, M_0 form a *uncertainty decreasing chain* with UNC uncertainty relation defined above: $M_4 >_{Mu} M_3 >_{Mu} M_2 >_{Mu} M_1 >_{Mu} M_0$. They also form a *generality decreasing chain* $M_4 > _{Mg} M_3 > _{Mg} M_2 > _{Mg} M_1 > _{Mg} M_0$. Here every model M_i can be obtained by specialization of parameters of model M_{i+1}, but we did not define the generality concept formally yet. Below it is done by using the parameterization approach. Each considered model has four parameters, p_1, p_2, p_3, and p_4. For instance, for model $M_2= ax^2+3y+7x+10$ parameter p_1=1represents uncertainty of ax^2, with an unknown coefficient a. Similarly, $p_2=p_3=p_4=0$, because further coefficients 3, 7 and 10 are known. Each model is represented as a Boolean vector,$\mathbf{v_i}$ $=(v_{i1},v_{i2},..,v_{ik},\ldots, v_{in})$: M_4: v_4=1111; M3: v_3=1110; M_2: v_2=1100; M_1: v_1=1000; M_0: v_0=0000.

Definition. Parametric model M_i is ***no less general*** than model M_j if $v_i \geq v_j$, i.e., $\forall k\ v_{ik} \geq v_{jk}$.

In accordance with this definition we have $1111 \geq 1110 \geq 1100 \geq 1000 \geq 0000$, that is isomorphic to $M_4 > _{Mg} M_3 > _{Mg} M_2 > _{Mg} M_1 > _{Mg} M_0$.

Now we apply learning operator $C(M_j,E)$ and produce a chain of models, where each model M_{j+1} is more specific then model M_j with decreasing parameters p_{i2}.

$$M_n > _{Mg} M_{Mn-1} \ldots M_{j+1} > _{Mg} M_j \ldots M_2 > _{Mg} M_1 > _{Mg} M_0 .$$

Each of these models has n+m parameters. We already assumed that n parameters p_{i1} are known. Encode known parameters as 1 and unknown as 0, to get a n+m-dimensional Boolean vector for model M_0 : $\mathbf{v}_0 = (v_{01}, v_{02},..,v_{0k},\ldots, v_{0n+m}) = (1010\ldots.10)$. In this notation we represent each model for n+m=6: M_3: v_2=111111; M_2: v_2=011111; M_1: v_1=011110; M_0: v_0=101010 and $111111 \geq 011111 \geq 011110 \geq 101010$ that is consistent with $M_3 > _{Mg} M_2 > _{Mg} M_1 > _{Mg} M_0$. A more detailed uncertainty parameterization can be developed if Boolean vectors are substituted by k-valued vectors $\mathbf{u_i} = (u_{i1}, u_{i2},.., u_{ik},\ldots,u_{in+m})$ with $u_{ij} \in U=\{0,1/(k-1),2/(k-1),\ldots k-2/(k-1),1\}$.

Definition. Parametric model M_i is ***no less general*** than model M_j if $\mathbf{u_i} \geq \mathbf{u_j}$, i.e., $\forall k\ v_{ik} \geq v_{jk}$.

Above we encoded known parameters as 1 and unknown as 0. Now we can assign a level of parameter uncertainty u_{i2} by computing $p_{i2}(Mj)/p_{i2max}$ and assigning u_{i2} as a nearest number from $\{0,1/(k-1),2/(k-1),\ldots k-2/(k-1),1\}$.

V. SIMILARITY MAXIMIZATION

A similarity maximization problem is a major mechanism of the dynamic logic that is formalized below.

Definition. A similarity L_{fin} measure is called a **final similarity measure** if $\forall$ M, E, L_i $L_i(M, E) \geq_{Lu} L_{fin}(M, E)$ The final similarity measure sets up the level of certainty of model similarity to the data that we want to reach.

Definition. The **static model optimization problem (SMOP)** is to find a model M_a such that

$$L_{fin}(M_a, E)=Max_{\ i\in I}\ L_{fin}(M_i,E) \qquad (1)$$

subject to conditions (2) and (3):

$$\forall\ M_j\ \ L_{fin}(M_a,E) =L_{fin}(M_j, E)\ \Rightarrow\ M_a \geq_{Mu} M_j\ , \qquad (2)$$

$$\forall\ M_j\ \ ((L_{fin}(M_a,E) =L_{fin}(M_j, E)\ \&\ ((M_j \geq _{Mg} M_a)\ v$$

$$((M_a \geq _{Mg} M_j)))\ \Rightarrow\ M_a \geq_{Mg} M_j \qquad (3)$$

Conditions (2) and (3) prevent model overfitting and beneficial computationally if further specification of the model requires more computations. Condition (2) means that if M_a and M_j have the same similarity measure with E, then uncertainty of M_a should be no less than uncertainty of M_j. Condition (3) means that if M_a and M_j have the same similarity measure with E and M_j and M_a are comparable relative to generality relation $\geq _{Mg}$ then M_a should be no less general than M_j. M_j can be obtained by specification of M_a.

Definition. **The dynamic logic model optimization (DLPO)** problem is to find a model M_a such that

$$L_a(M_a, E)=Max_{\ i\in I}\ L_i(M_i,E) \qquad (4)$$

subject to conditions (3) and (4):

$$\forall\ M_j\ \ L_a(M_a,E) =L_j(M_j, E)\ \Rightarrow\ M_a \geq_{Mu} M_j\ , \qquad (5)$$

$$\forall\ M_j\ \ ((L_{fin}(M_a,E) = L_j(M_j, E)\ \&\ ((M_j \geq _{Mg} M_a)\ v$$

$$((M_a \geq _{Mg} M_j)))\ \Rightarrow\ M_a \geq_{Mg} M_j \qquad (6)$$

This is a non-standard optimization problem. In the standard one, only models M_i are changed but the optimization criterion L is not. It does not depend on the model M_i. In DL, the criterion L is changing dynamically with models M_i. Since the focus of NMF is cutting **computational complexity** (CC) of model optimization, a **dual optimization problem** can be formulated.

Definition. Mapping C: $\{M\}\rightarrow\{M\}$ is called a **learning (adaptation) operator** C, $C(M_i,E)= M_{i+1}$, where E are data and $M_i \geq_{Mu} M_{i+1}$, $M_i \geq_{Mg} M_{i+1}$.

This operation represents a *Cognitive learning process* C of a new model M_{i+1} from a given model M_i and data E. In other words it is an adaptation of model M_i to data E that produce model M_{i+1}.

Definition. An optimization problem of finding a **shortest sequence** of matched pairs (M_i , L_i) of models M_i and optimization criteria (similarity measures) L_i that solves the optimization problem (4)-(6) for the given data E is called a **dual dynamic logic model optimization (DDLMO)** problem, which finds a sequence of n matching pairs

$$(M_1 , L_1),\ (M_2 , L_2),\ldots, (M_n , L_n),$$

such that $L(M_n,E)= \text{Max}_{\ i\in I}\ L(M_i,E)$ and $\forall\ M_i\ L_i=F(M_i)$, $C(M_i,E)= M_{i+1}$, $M_i \geq_{Mu} M_{i+1}$, $M_i \geq_{Mg} M_{i+1}$, $M_n =M_a$, $L_n=L_a$.

This means finding a sequence of more specific and certain models for the given data E, matching operator F and learning operator C that maximizes $L(M_i,E)$.

VI. MONOTONE BOOLEAN FUNCTIONS

Definition. A Boolean function f: $\{0,1\}^n \rightarrow \{0,1\}$ is a **monotone Boolean function** if: $\mathbf{v}_i \geq \mathbf{v}_j\ \&\ \Rightarrow\ f(\mathbf{v}_j) \geq f(\mathbf{v}_i)$.

This means that ($\mathbf{v}_i \geq \mathbf{v}_j$ & $f(\mathbf{v}_i)=0$) $\Rightarrow$ $f(\mathbf{v}_j)=0$ and ($\mathbf{v}_i \geq \mathbf{v}_j$ & $f(\mathbf{v}_j)=1$) $\Rightarrow$ $f(\mathbf{v}_i)=1$. Function f is non-decreasing. Consider fixed E and M_i that are parameterized by $\mathbf{v}_i$ and interpret $L(M_i,E)$ as $f(\mathbf{v}_i)$, i.e., $L(M_i,E)=f(\mathbf{v}_i)$. Assume that $L(M_i,E)$ has only two values (unacceptable 0, and acceptable 1). It can be generalized to a k-value case if needed. If $L(M_i,E)$ is monotone then $\mathbf{v_i} \geq \mathbf{v_j}$ $\Rightarrow$ $L(M_i,E)) \geq L(M_j,E)$, e.g., if $L(M_{3\text{-}1110},E)=L(M_{2\text{-}1100},E)=0$ then

$$(\mathbf{v}_i \geq \mathbf{v}_j\ \&\ L(M_{3\text{-}1110},E)=0) \Rightarrow\ L(M_{2\text{-}1100},E)=0 \qquad (7)$$

$$(\mathbf{v}_i \geq \mathbf{v}_j\ \&\ L(M_{2\text{-}1100},E)=1) \Rightarrow\ L(M_{3\text{-}1110},E)=1 \qquad (8)$$

This means that if a model with more unknown parameters $\mathbf{v}_i$ failed then a model with less unknown parameters $\mathbf{v}_j$ will also fail. If we conclude that a quadratic polynomial model (M_2) is not acceptable, $L(M_2,E)=0$, then a more specific quadratic model M_3 also cannot be acceptable, $L(M_3,E)=0$. Thus, we do not need to test model M_3. This monotonicity idea helps to decrease computational complexity. We use

$$(\mathbf{v}_i \geq \mathbf{v}_j\ \&\ f(\mathbf{v}_i)=0) \Rightarrow\ f(\mathbf{v}_j)=0 \qquad (9)$$

for rejecting models and

$$(\mathbf{v}_i \geq \mathbf{v}_j\ \&\ f(\mathbf{v}_j)=1) \Rightarrow\ f(\mathbf{v}_i)=1 \qquad (10)$$

for confirming models. In the case of **model rejection test** for data E, the focus is not quickly building a model but quickly rejecting a model M (Popper's principle). In essence the test $L_3(M_3,E)=0$ means that the whole class of the models M_3 with 3 unknown parameters fails. For testing M_3 positively for data E we need to find 4 correct parameters. This may mean searching in a large 4-D parameter space $[-100,+100]^4$ for single vector, say $(p_1,p_2,p_3,p_4) = (9, 3, 7, 10)$, if each parameter varies in the interval [-100,100]. For rejection we may need only, 4 training vectors (x,y,u) from data E and 3 test vectors. The first four vectors allow us to build a quadratic surface in 3-D as a model. We just test that three test vectors from E do'nt fit this quadratic surface.

VII. SEARCH PROCESS

In the optimization process, we want to keep track of model rejections and to guide dynamically, what model will be tested next to minimize the number of tests. Formulas (9) and (10) are key formulas to minimize tests, but we need the whole strategy how to minimize the number of tests and formalize it. It can be done by minimizing **Shannon function** φ [1] $\min_{\ A\in\mathbf{A}} \max_{\ f\in\mathbf{F}}\ \varphi(f,A)$, where **A** is a set of algorithms, F is a set of monotone functions and φ(f,A) is a number of tests that algorithm A does to fully restore function f. Each test means computing a value f(**v**) for a particular vector **v**. In the theory of monotone Boolean functions it is assumed that there is an **oracle** that is able to produce the value f(**v**), thus each test is equivalent to a request to the oracle [1,4,5]. Minimization of the Shannon function means that we search for the algorithm that needs the smallest number of tests for its worst case (function f that needs maximum number of tests relative to other functions). This is a classic min-max criterion. It was proved in [1] that

$$\min_{\ A\in\mathbf{A}} \max_{\ f\in\mathbf{F}}\ \varphi(f,A) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor (n/2 \rfloor + 1},$$

$\lfloor x \rfloor$ is a floor of x (an integer that is smaller than x and closest to x). The proof is based on the structure called Hansel chains. These chains cover the whole n-dimensional binary cube $\{1,0\}^n$. The steps of the algorithm are presented in detail in [3,5]. The main idea of these steps is building Hansel chains, starting from testing his smallest chains, expanding each tested value using (a) and (b) formulas presented above and test values that are left not expanded on the same chains then move to larger chains until no chains left. The goal of the search is to find a **smallest lower unit v**, i.e., a Boolean vector such that f(**v**)=1, and for every w<v f(w)=0, and for every u>v |u|>|v|. A simpler problem could be to find any lower unit of f.

The search problem in logic terms can be formulated as a satisfiability problem: Find a system of axioms T_a such that $L_a(T_a, E)=1$ subject to the condition $\forall\ T_j\ L_a(T_a,E) =L_j(T_j, E)$ $\Rightarrow\ T_j \geq_{Mu} T_a$, i.e., if T_a and T_j have the same similarity with E, then M_a should have a lower uncertainty than T_j, e.g., $T_j \geq_{Mu} M_a$. If a similarity measure is defined as a **probabilistic measure** in [0,1] then the probabilistic version of the task of finding system of axioms T for model A that maximizes a probabilistic similarity measure is: $\text{Max}_{\ i\in I}$ $L(M_i,E)$, where $L=F(M_i)$ and I is a set of models.

VIII. FORMAL DESCRIPTION OF P-DYNAMIC LOGIC

Below we describe formally our dynamic logic of models of phenomena (**P-dynamic logic** for short) as a summarization of concepts introduced above. As any logic, this logic consists of three parts: (1) a syntactical symbolic part, (2) a reasoning part, and (3) a semantic part.

A. Semantic part of P-dynamic logic

At first, we define **semantic part** of P-dynamic logic. It consists of two related algebraic systems. The first one is as a *three-sort algebraic system* [7] of signature Ω, where {E} are sets of data, {M} are sets of P-models, R is the set of real numbers,

$$\mathbf{EM} = < \{E\}, \{M\}, R; \Omega_{EM}>. \quad (11)$$

The signature Ω_{EM} consists of sets of relations Ω_E, Ω_{PM}, Ω_R on {E}, {M**}** and R, respectively, and operators {L} and C that connect {E}, {M} and R,

$$\Omega_{EM} = <\Omega_E, \Omega_{PM}, \Omega_R, \{L\}, C >. \quad (12)$$

The signature of P-models $\Omega_{PM} = \{ \geq_{Mg}, \geq_{Mu}, \geq_{Ms} \}$ presents order relations between P-models relative to their generality, uncertainty and simplicity described above. Each **similarity (correspondence) measure** $L_i \in \{L\}$ is a mapping:

$$L_i: \ \{M\}\times\{E\} \rightarrow R \quad (13)$$

that captures numerically the similarity between data and P-model. The higher value of $L_i(M,E)$ indicates the higher consistency between data E and the P-model M in the aspect that was captured by L_i. Having a set of such measures {L} allows us to choose them dynamically.

Next, a P-model **enhancement (learning) operator** is

$$C: \ \{ M, E\} \rightarrow \{M\} \quad (14)$$

that changes a P-model. This operator brings us dynamics of model changes. Thus, we have two types of dynamics and we need a formal way to express the change of L similarly to (14) for models. We do this by introducing a *two-sort algebraic system*

$$\mathbf{ML} = < \{M\}, \{L\}; \Omega_{PM}, \Omega_L, F >, \quad (15)$$

with the signature that includes

$$\Omega_{PM} = \{ \geq_{Mg}, \geq_{Mu}, \geq_{Ms} \}, \quad \Omega_L = \{ \geq_{Lg}, \geq_{Lu} \}$$

and F as a mapping between sets {M**}** and {L**}** that preserves relations on {M},

$$F: \ \{M\} \rightarrow \{L\} \quad (16)$$

Algebraic systems (11) and (15) have several common components, but {L} has very different roles in them. In (11) {L} is part of the signature Ω, i.e., is it is an *operator*, but in (15) {L} is a one of two *base sets* of the algebraic system, where the operator is F: {M} → {L}.

Thus, we cannot simply join systems (11) and (15) together to a single algebraic system having different roles of {L} in (11) and (15). We would need a generalized multi-sort algebraic system in the second order logic to do this. Note that, separately (11) and (15) are much simpler systems in the first order logic (FOL).

In addition, we can build (15) without a specific dataset E∈{E}, because (15) does not require {E}. For constructing (11) for a specific dataset E, we would need to define Ω_E. See an extensive discussion of these issues and examples in [6,4]. Thus the semantic part of the P-dynamic logic is

$$\mathbf{EMML} = <\mathbf{EM}, \mathbf{ML;} W >$$

where $W(M_i, E, M_j)$ is the following relation between **EM** and **ML:**

$$W(M_i, E, M_j) \equiv$$

$$\{ [C \in \Omega_{EM} \ \& \ C(M_i,E) = M_j \ ] \ \& \quad (17)$$

$$\{[(M_i >_u M_j) \vee (M_i >_g M_j) \vee (M_j >_s M_i)] \vee \quad (18)$$

$$[L_i = F(M_i) \ \& \ L_j = F(M_j) \ \& \ L_j(M_j,E) > L_i(M_i,E)]\}) \quad (19)$$

Relation $W(M_i, E, M_j)$ is true if and only if C produces P-model M_j using E that is better than input P-model M_i in at least one of its characteristics (more certain, more specific, simpler, or better fit data relative to similarity measures). In other words, a Boolean predicate $W(M_i, E, M_j)=1$ if $C(M_i,E) = M_j$ produced an improved model M_j, else $W(M_i, E, M_j)=0$.

In accordance with definitions in previous sections, in (18) $M_i >_u M_j$ means that M_j is a more certain model than M_i. Similarly, $M_i >_g M_j$ means that M_j is a more specific, and $M_j >_s M_i$ means that M_j is simpler than M_i. Property (19) means that model M_j better fits data E than model M_i relative to measures L_j and L_i which are dynamically assigned to M_j and M_i by applying F to them.

Relation W sets up a semantic criterion for C to be a learning operator for P-models M_i, M_j and data E not only to be called a learning operator, simply because it produces another model. Note that (17) and (18) in W can be checked having only **EM** but (19) requires **ML** too.

Now we want to discuss how to compute truth-values of predicates and outputs of operators in **EM** and **ML**. We need to analyze all of them to answer this question.

For **EM** and **ML** we gave some examples of Ω_M and Ω_L, {L}, C and F. An extensive set of examples of similarity measures L and learning operators C are given in [8]. For ML in addition we need to compute F. For instance if M(c,r) is a model that represents a circle with center c and radius r then L(c,r) could be the same, F(M(c,r)) = L=M(c,r). This means that L tests exactly model M(c, r). An alternative F could be F(M(c,r))= L=M(c, r+e) that accepts all models with radiuses no greater than r+e.

In general a library of such matching operators F and relations Ω_M and Ω_L should be created that will be available to researchers. Thus, we have a semantic part of the P-dynamic logic identified.

B. Syntactic part of P-Dynamic Logic

Now we need to explore which part of this semantic part can be transferred to the syntactic level so that we could do reasoning without semantic knowledge to get at least some non-trivial inferences and conclusions.

To distinguish syntactic level from semantic one we will use low-case notation **em** and **ml** to define syntactic **EM** and **ML**. We do this for all components thus,

$$\mathbf{em} = < \{e\}, \{m\}, r; \omega_{em}> \quad (20)$$

The signature ω_{em} consists of sets of relations ω_e, ω_m, ω_r on {e}, {m**}** and r, respectively, and operators {l} and c that connect {e}, {m} and r,

$$\omega_{em} = <\omega_e, \omega_m, \omega_r, \{l\}, c > \quad (21)$$

Similarly, we define

$$\mathbf{ml} = < \{m\}, \{l\}; \omega_m, \omega_l, f >, \quad (22)$$

with the signature that includes

$$\omega_m = \{ \geq_{mg}, \geq_{mu}, \geq_{ms} \}, \omega_l = \{ \geq_{lg}, \geq_{lu} \}\ f: \{m\} \rightarrow \{l\} \quad (23)$$

In the same way we define **emml** = <**em**, **ml;** w >, where

$$w(m_i, e, m_j) \equiv$$
$$([c \in \omega_{em} \ \& \ c(m_i,e) = m_j] \Leftrightarrow \quad (24)$$
$$\{[(m_i >_{mu} m_j) \vee (m_i >_{mg} m_j) \vee (m_j >_{ms} m_i)] \vee \quad (25)$$
$$[l_i = f(m_i) \ \& \ l_j = f(m_j) \ \& \ l_j(m_j,e) > l_i(m_i,e)]\}) \quad (26)$$

C. Reasoning part of P-dynamic logic

Now we will discuss a reasoning part of P-dynamic logic. To clarify this issue we need to answer the following question: "What could be the most interesting part of the syntactic reasoning for practical use?" Assume that we already have a knowledge base (KB). This KB contains $m_1,\ldots m_n$, and e called facts in this KB ($m_{1,\ldots}m_n$ are interpreted semantically as P-models and e as data). KB also contains some expressions, e.g., $w(m_i, e, m_j)$ (interpreted semantically as P-model m_j is an improved P-model m_i).

We want to know if $w(m_i, e, m_k)$ can be inferred or not knowing that $w(m_i, e, m_j)$ and $w(m_j, e, m_k)$ are in the KB without using a semantic interpretation of them pure syntactically. Having this we will have useful syntactic reasoning in this DL.

It follows from semantics of relation W that it is transitive, thus we can postulate transitivity for w. As a result of this postulate we can infer $w(m_i, e, m_k)$ pure syntactically having $w(m_i, e, m_j)$ and $w(m_j, e, m_k)$ in KB without going to the semantic level and computing $W(M_i, E, M_k)$ which can be computationally challenging for a large dataset E. This is a major advantage of using P-DL syntactic reasoning instead of computations at the semantic level.

The reasoning mechanism T_{em} in **em** is first order logic with terms in its signature. Similarly, T_{ml} in **ml** it is first order logic in terms of its signature. In <**em**, **ml;** w > we have T_{emml} that is also a first order logic reasoning with w, but if we substitute w with its components (24)-(26) we will have a second order logic reasoning. Thus the complete description of the **dynamic logic of phenomena models** is

$$DL = < DL_{EM}, DL_{ML}, DL_{EMML} >$$

where

$$DL_{EM} = < em, T_{em}, \mathbf{EM} >,$$

$$DL_{ML} = < ml, T_{ml}, \mathbf{ML} >,$$

$$DL_{EMML} = < \mathbf{emml}, T_{emml}, \mathbf{EMML} >.$$

D. Learning operator

Now we want to elaborate a concept of the learning operator, C. There is an important questions about this operator: "How sophisticated C should be relative to a brute force algorithm that computes L(M,E) for every P-model M and selects M that provides min L(M,ER)?"

It will be very advantageous to get a simple and quite universal operator C, which will allow us to use it for solving a wide variety of problems. Let us consider a direct modification of brute force algorithm to the situation with dynamic change of correspondence (similarity) measures L_i, which is assumed in the dynamic logic. The next assumption is that the space of highly uncertain models is relatively small. Thus, a brute force algorithm can work for these models in a reasonable time. Next, the best model M_1 found at this step will produce a new correspondence measure L_1 and this measure will be applied to a *new set of models* produced by M_1 for evaluation (e.g., with a more dense grid around M_1 or in another location if $L(M_1)$ is low). To produce new models we introduce a new operator, H, that we will call a **specialization operator**, as follows:

Step 1. Select initial P-model M_0.
Step 2. Produce set of P-models $H(M_0)$
Step 3. Compute $L_0(M, E)$ for every M from $H(M_0) = \{M\}$ and find model $M_1 = \arg\min_{\{M\}} L_0(M, E)$.
Step 4. Test if $L_0(M_1,E) > T$, i.e., is above the needed correspondence threshold. Stop it is true, else go to step 5.
Step 5. Repeat steps 1-4 until all models tested or time limit reached.

A more complex strategies for H operator can be based on breadth first, depth first, and branch-and-bound strategies

E. Cognitive Dynamic Logic

Now we can generalize Phenomena Dynamic logic (P-DL) to a **Cognitive Dynamic logic (C-DL)**. We follow conceptual framework outlined in [11] and introduce, order relation "$>_c$" which is applied to objects that we call **cognitive models (C-models**). These models are P-models discussed above with an additional relation "$>_c$" in model M_j is no less **conscious** than model M_i or model M_i is no more conscious than model M_j. The relation $M_i \geq_{Mg} M_j$ has the same meaning as we first introduced it, M_i is no less *general* than M_j" or "M_i no less *abstract* than M_j" and respectively M_i is no more *specific* or *concrete* then M.. The relation $\geq_{Mu}$ expresses that one model in no less *uncertain (*no less *vague* or no less *fuzzy)*.

IX. CONCLUSION AND FUTURE WORK

This paper has introduced a formalization of the dynamic logic in the terms of the first order logic, logic model theory and theory of Monotone Boolean functions. The formalization covers the main idea of DL -- matching levels of uncertainty of the problem/model and levels of uncertainty of the evaluation criterion, which dramatically decrease computation complexity of finding a model.

The core of formalization is a partial order on the phenomena models and similarity measures with respect to their uncertainty, generality, and simplicity. For cognitive

models, we added a partial order of model consciousness. These partial orders are represented using a set of Boolean parameters and visualized using the theory of monotone Boolean functions to monitor and guide the search in the Boolean parameter space in the DL setting.

Further theoretical studies may reveal deeper links with classical optimization search processes and significantly advance them by adding an extra layer of constructing optimization criteria in addition to using them. In the area of logic, further studies may reveal deeper links with classical logic problems such as decidability, completeness, and consistency. This work also gives a new perspective for machine learning [2,4] to develop learning algorithms that can learn evaluation criteria and models simultaneously. We expect that links with another Dynamic Logic [12] motivated by logical analysis of computer programs will also be established.

The proposed formalization creates a framework for developing specific applications and modeling tools. We envision a modeling tool that will consist of sets of models, matching similarity measures, processes for testing them and model learning processes for specific problems in pattern recognition, data mining, optimization, cognitive process modeling, and decision making.

X. Appendix: Computational complexity

Below we describe two examples that give computational motivation to our research on building a logic formalism of the phenomena dynamic logic. The first example is about finding a hidden circle in the noisy data when the level of noise is so high that a direct human observation does not allow getting any visual clue where the circle is located in the image. In the second example, the shape of the object to be found is more complex (see Fig. 1 on the right). To identify the circle we need two points C and H, and to identify the second (parabolic) shape we need 4 points A,B,D and G (see Fig. 1).

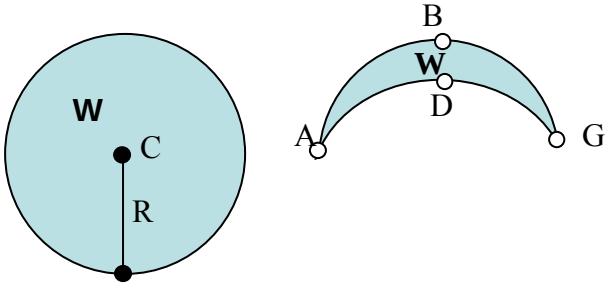

Figure 1. Modeled shapes

To solve the first example we assume:

1. The hidden shape to be discovered is a circle.
2. The density of the points in the circle is *higher* than its signature, $\Omega_{CM} = \{ \geq_{Mg}, \geq_{Mu}, \geq_{Ms}, \geq_{Mc} \}$.
3. The semantic interpretation of relation $M_j \geq_c M_i$ is that outside of the circle W.
4. The *center* c=(x_c, y_c) of the circle *is not known*, but its range is known: $x_c \in [x_{min},x_{max}]$, $y_c \in [y_{min},y_{max}]$.
5. The *resolution* of x and y *are known*, e_x and e_y, that is x has a finite set of n_x values, x_{min}, $x_{min}+e_x,\ldots$, $x_{min}+ke_x,\ldots x_{max}$ and y has a finite set of n_y values, y_{min}, $y_{min}+e_y,\ldots,y_{min}+re_y,\ldots y_{max}$
6. The *radius* R of W *is not known*.
7. The radius R resolution is $e_r = \min(e_x,e_y)$.
8. More than one circle can be in the area, the *number* of the circles *is not known*.
9. Different circles may or may not overlap.
10. The circles' area is less than 1/3 of the whole area.

The brute force algorithm for the example 1 is:

```
For (int k = 0; k < n_x ;k ++)                  // x loop
    {  For (int r = 0;  r <n_y  ;r ++)          // y loop
         {  For (int t =0; t < max(n_x,n_y) ;t ++)  // radius loop
                  { DensityDifferenceTest(x_c,y_c,r )}
  // return 1 if D(R) – D(¬R) >T, else 0, that is if density
  // D(W) in circle W is greater than D(¬W) with threshold
  // T, where D(¬W) is density outside W,
  // C=( x_c,y_c) is circle's center and R is its radius.
  }}
```

The **Computational Complexity** (CC) of this algorithm with the base operation as DensityDifferenceTest (x_c,y_c, R) is $n_x \cdot n_y \cdot \max(n_x,n_y)$. If $n_x = n_y = n$ then it is $\boldsymbol{O(n^3)}$**.** The CC of DensityDifferenceTest function depends on the total number *m* of given points

The brute force algorithm for DensityDifferenceTest function presented below uses the following notation: M is the number of input points, Q is the total area, T is density threshold, IPC inside point count (initiated 0), OPC is outside point count, INAR is the area of the circle, OUTAR is the area outside of the circle, Din is the density in the circle, and Dout is the density out of the circle

```
 DensityDifference (x_c, y_c, r ){
For (i=0; i < m; i++) {if InsidePointTest(i, R)=1 then {
     IPC=IPC+1,  OPC= m – IPC} // inside test and count
      Read (T);        // Enter threshold T
      Q=(x_max-x_min)·(y_max-y_min); // Total area
      INAR=Area(x_c, y_c, R); // Compute circle inside area
      OUTAR= Q – INAR; // Outside area
      Din=IPC/INAR;        // inside point density
      Dout=OPC/Dout;        // outside point density
      If (Din-Dout > T)  Return  1, x_c, y_c, R. }
```

For the circle InsidePointTest(i, R)=1 if $(x_i - x_c)^2+(y_i - x_c)^2 \leq R^2$, else 0. The computational complexity of this Density Difference Test algorithm is *m* with the base operation as testing $(x_i - x_c)^2+(y_i - x_c)^2 \leq R^2$, that is if the point is inside of the circle. Thus, the total computational complexity of the brute force algorithm is $\boldsymbol{O(n^3m)}$. If input points covers the whole grid then *m*=*n* and we have CC equal to $\boldsymbol{O(n^4)}$. If fact m is a fraction of n, thus if m=n/10 then we still have $O(n^4)$.

Now we will change our assumption 1 that the hidden shape to be discovered is a circle. A more complex shape to discover is produced by two quadratic curves shown in Fig.1. Three points A,B, and G are sufficient to identify the first curve and three points A,D,G are sufficient to identify

the second curve. Thus, total 4 points are sufficient to identify both curves. In the case of the circle, we need only two points, center C, and any point H on the circle, that gives us the radius. To solve the task for our new shape, we keep the same assumptions as we had for the circle. We assume the unknown number of shapes of unknown sizes, locations, and orientations.

The algorithm for *parabolic shape* is modified as follows:

```
For (int ax = 0; ax < nx ; ax ++)            // x loop for point A
  {For (int ay = 0;  ax <ny ;ax ++) // y loop for A
    For (int bx = 0; bx < nx ;bx ++) // x loop for B
      {For (int by = 0; by <ny; by ++) // y loop for B
          {For (int gx =0; gx < nx;gx ++) // x loop for G
            {For (int gy = 0;  gy <ny  ; gy ++)
                                        // y loop for G
              {For (int dx = 0; dx < nx ; dx ++)
                                        // x loop for D
                {For (int dy = 0; dy <ny ; dy ++)
                                        // y loop for D
                  { DensityDifferenceTest(A,B,G,D)}
  // return 1 if D(W) – D(¬W) >T, else 0, i.e., if density
  // D(W) in W is greater than outside with threshold T
  }}}}}
```

Complexity of this algorithm is $O(n^4)$ if n_x and n_y are equal to n. The density difference test has the same complexity $O(m)$ as above for the circle, but a more complex for points inside of the shape. However, this test time is growing linearly with m. For each point, it computes two quadratic forms and tests if the point is in or out of the shape based on these values. This time is limited by a constant for each given point. Thus the total complexity is $O(n^4m)$ and if m would reach n then it will be $O(n^5)$.

Below we tabulate the complexities found above to identify limits of practical use of the brute force algorithm. We assume that $n_x = n_y = n$ and $m = n/10$, thus our complexity function is $K_1 n^3 n/10 = K_1(n^4/10)$ for the circle and $(K_2 n^4)(n/10) = K_2 (n^5/10)$ for the new shape, where K_1 and K_2 are respectively time to compute the base operation for the circle and the new parabolic shape. We also assumed 10^9 *base operations/sec* in these computations. Note that this base operation is a part of the Density Difference Test computed for each of m inputs and takes more time than a base processor operation. Tables 1 and 2 show computational complexity under these assumptions for search of circles and parabolic shapes. A computer screen grid of pixels is about 1000×1000 and images from current cameras are larger. Only for a single screen with 100 input points finding the circle has an acceptable time (100 sec) which can be too slow for real time applications. Finding a more complex parabolic shape needs 28 hours for the same 100 input points. Increasing m to 1000 or 10000 input points leads to many years as table 2 shows.

Real objects such as planes have shapes that are more complex. Thus, more critical points are needed to identify their models and CC will be greater. Therefore, fundamentally different approaches are needed to deal with such huge computational complexity issues.

This is a motivation of this paper -- building the dynamic logic formalism for the dynamic logic process.

A. Exponential complexity case

Above we considered a situation where we need to test a polynomial number of models. This was a result of our assumption that models are circles or parabolic shapes defined by n^2 points on the grid that need 2-4 parameters also determined by the same grid. If objects are more complex (e.g., long and non-linear tracks) then their models are more complex shapes. Moreover, if models are not based on such few parameters but need to be identified by exploring all possible subsets of m then we will have 2^m tests, that is of exponential computational complexity and needs more time than shown in Tables 1 and 2.

TABLE 1.
COMPUTATION TIME FOR SEARCH OF CIRCLES IN NOISY DATA

Grid size n	*Points m*	*Circle*	
		Computational Complexity n^3m	*Time for 10^9 base operations per sec*
10	1	1000	0.000001 sec
100	10	1E+7	0.01 sec
1000	**100**	1E+11	**100 sec**
10000	**1000**	1E+15	**277.8 hours**
100000	10000	1E+19	126.8 years
1000000	100000	1E+23	1,27 million years

TABLE 2.
COMPUTATION TIME FOR SEARCH OF PARABOLIC SHAPES IN NOISY DATA

Grid size n	*Points m*	*Parabolic shape*	
		Computational Complexity n^4m	*Computational Complexity n^4m*
10	1	10000	0.00001sec
100	10	1E+09	1 sec
1000	**100**	1E+14	**28 hours**
10000	**1000**	1E+19	**317.1 years**
100000	10000	1E+24	31.7 million years
1000000	100000	1E+29	3.17E+12 years

REFERENCES

[1] G. Hansel, G., Sur le nombre des fonctions Boolenes monotones de n variables. *C.R. Acad. Sci*. Paris, v. 262, n. 20, 1088-1090, 1966.

[2] Luc De Raedt. *From Inductive Logic Programming to Multi-Relational Data Mining*. Springer, 2006

[3] B. Kovalerchuk, F. Delizy, "Visual data mining using monotone Boolean functions", In: *Visual And Spatial Analysis: Advances In Data Mining, Reasoning, And Problem Solving*, Kovalerchuk, B, Schwing, J, (Eds), Springer, 2005, pp. 387-406.

[4] B. Kovalerchuk, Vityaev E., *Data Mining in Finance: Advances in Relational and Hybrid Methods*, Kluwer, 2000.

[5] B. Kovalerchuk, Triantaphyllou, E., Despande, A., Vityaev, E., "Interactive Learning of Monotone Boolean Function". *Information Sciences*, Vol. 94, 1-4, 1996, pp. 87-118

[6] D. H. Krantz, R. D. Luce, P. Suppes, A. Tversky. *Foundations of Measurement*. New York, London: Academic Press, 1971-1990

[7] A.Malcev. *Algebraic Systems*. Springer-Verlag, 1973

[8] L. Perlovsky, *Neural Networks and Intellect: Using Model-Based Concepts*, Oxford University Press, 2000

[9] L. Perlovsky, "Toward physics of the mind: Concepts, emotions, consciousness, and symbols", *Physics of Life Rev.* 3, 2006, pp 23–55.

[10] W. Hodes, First-order Model Theory, Stanford Encyclopedia of Philosophy, 2005, http://plato.stanford.edu/entries/modeltheory-fo/

[11] L. Perlovsky, "Evolution of language, Consciousness, and Cultures", *IEEE Computational Intelligence Magazine*. 8, 2007, pp. 25-39.

[12] D. Harel, D. Kozen, and J. Tiuryn, *Dynamic Logic*, MIT Press, 2000.