Representation Before Computation

Lev Goldfarb

Inductive Information Systems
Fredericton, NB
Canada
lev.goldfarb@gmail.com
http://www.cs.unb.ca/~goldfarb

Abstract. My main objective is to point out a *fundamental* weakness in the conventional conception of computation and suggest a promising way out. This weakness is directly related to a gross underestimation of the role of object representation in a computational model, hence confining such models to an unrealistic (input) environment, which, in turn, lead to "unnatural" computational models. This lack of appreciation of the role of *structural object representation* has been inherited from logic and partly from mathematics, where, in the latter, the centuries-old tradition is to represent objects as unstructured "points". I also discuss why the appropriate fundamental reorientation in the conception of computational models will bring the resulting study of computation closer to the "natural" computational constrains. An example of the pertinent, class-oriented, representational formalism developed by our group over many years—Evolving Transformation System (ETS)—is briefly outlined here, and several general lines of research are suggested.

1 Introduction

Historically, computability theory has emerged within logic [1], and so it is not surprising that "logical" agenda continues to play an important part in its development [2]. This origin has determined to a considerable extent the basic "logical orientation" of the field, i.e. the kinds of questions that one might be asking (computable vs. non-computable, etc.). [3]

[1] E.g. Alan Turing proposed the Turing machine as a convenient mechanism for answering question of whether satisfiability in the predicate calculus was a solvable problem or not.

[2] E.g. Stephen Cook [1]: "When I submitted the abstract for this paper in December 1970 my interest was in predicate calculus theorem proving procedures, and I had not yet thought of the idea of NP completeness."

[3] In connection with this orientation of the field, it is useful to recall ideas of John Von Neumann, one of the leading prewar logicians who was well aware of the original Turing work and actually invited Turing to stay at the Institute for Advanced Studies as his assistant (after Turing completed his Princeton dissertation): "We are very far from possessing a theory of automata which deserves that name, that is, a purely

At the same time, already in his original paper, Turing had to face the issue of representation, both the "data" representation (on the tape), i.e. the input "symbol space" [4], and the representation of the machine itself (machine instructions and its states).

Following the original Turing's considerations, related to "the process of [symbol] recognition", it is not difficult to see that if these "symbols" are endowed with a more general structure—as is the case with all real-world "symbols"—the structure of the corresponding "generalized Turing machine", or "intelligent machine", would have to be fundamentally modified in order to be able to deal with such structured symbols in an appropriate manner. So, first of all, we are faced with the ubiquitous issue of *structural representation*. I would like to emphasize that, when dealing with natural objects, e.g. molecules or flowers, the treatment of the issues related to the structural object representation must precede the computational considerations, since the formal structure of the latter should depend on the formal structure of the former: the structure of object operations will determine the structure of the corresponding basic "machine" operations. Perhaps an analogy may help. If we recall the concept of a mathematical structure (e.g. group, vector space, topological space, partially ordered set), we should favor the idea that the *structure of the machine* operating on the *data that is viewed as an element of a particular (fixed) mathematical structure* should depend on the underlying operations postulated within this mathematical structure. In other words, representational considerations should precede computational ones.

However simple the above considerations may look, the looks are deceiving: it turns out that so far mathematics (and logic) has not adequately addressed the issue of structural object representation, for good reasons. I suggest that these reasons are quite profound and have to do with the context within which the development of structural representation should take place. This context, of *classes and induction*, has never been adequately delineated neither within

---

mathematical-logical theory. There exists today a very elaborate system of formal logic and specifically, of logic as applied to mathematics. ... About the inadequacies [of logic], however, this may be said: Everybody who has worked in formal logic will confirm that it is one of the technically most refractory parts of mathematics. The reason for this is that it deals with rigid, all-or-none concepts .... The theory of automata, of the digital, all-or-none type ... is certainly a chapter in formal logic. It would, therefore, seem that it will have to share this unattractive property of formal logic."[2]

[4] "If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrary small extent. ... The new observed squares must be immediately recognizable by the computer. ... Now if these squares are marked only by single symbols there can be only a finite number of them, and we should not upset our theory by adjoining these marked squares to the observed squares. If, on the other hand, ... [each of them is] marked by a sequence of symbols [i.e. by a *structured* symbol], we cannot regard *the process of recognition* as a simple process. This is a fundamental point and should be illustrated. [He then considers symbols formed by numeric sequences.]"[3, my emphasis]

philosophy—in spite of continuous attention it attracted over many centuries—nor within psychology, artificial intelligence, and information retrieval.

This paper is motivated by a recent development of what should be called the first formalism for structural representation, the Evolving Transformation System (ETS). The main reason for the adjective "first" has to do with the following considerations, related to the above context of classes and induction: these ubiquitous concepts, for the first time, receive full clarification within the formalism, while, despite the common usage, strings and graphs cannot be considered as satisfactory forms of structural representation. Briefly, a string (or a graph) does not carry *within itself* enough representational information to allow for the inductive recovery of the corresponding class representation, i.e. grammar, which, as it turns out, is a serious indication of the inadequacy of the underlying representational formalism. Not surprisingly, the development of such a radically different formalism as ETS is still in its initial stages, and hence, in this paper, it is prudent to focus on the overall directions of its development rather than on any concrete formal results.

I should also mention that the above reorientation in the conception of a computational model is expected to bring it closer to a (structural) generalization of the conventional mathematical structures, in which the basic formal/underlying structure is postulated axiomatically.

2 On a Proper Approach to the Concept of Structural Object Representation

Historically, mathematics has been the main, if not the only, "science of representation", so it is not surprising that the lack of appreciation of the role of object representation has been inherited from mathematics, where the centuries-old tradition (including vector spaces) has been to represent objects as "unstructured" points. In this sense, the underestimation of the role of object representation in a computational model is also not surprising. But while the mainstream mathematics—having been somewhat isolated from the problems arising in computer science—had no *apparent* impetus to proceed with the necessary radical developments, computer science has no such excuse. The concepts of data structure and abstract data type have been "screaming" for such fundamental developments. As was mentioned in Introduction, I believe the reasons for the lack of such developments within computer science have to do with its historical roots in mathematical logic, whose development was not at all motivated [5] by the representation of "physical" objects [4], [5].

First, I propose to proceed with the development of formalism for structural representation via generalization of the most basic entities on which the entire edifice of mathematics and natural sciences stands, i.e. via structural generalization of the natural numbers: at this stage, I don't think we have a more

[5] I.e. considerably less so than is the case with mathematics, which always *had to address* the concerns of physics.

fundamental choice for our starting point. I do not have in mind here relatively "minor" (historical) generalizations such as complex numbers, quaternions, etc., since such generalizations are still numeric-based. What I have in mind is more far-reaching generalization, based on the generalization of the Peano *constructive process* in which a single "unstructured" successor operation is replaced by several structural ones: I am thinking of the entities that are structured in a manner shown in Figs. 1, 2.

Second, I suggest that the notion of structural representation can properly be addressed only within the context of a class. The reasons have to do with the following hypothesized ontology of objects. As we know, objects in nature do not pop up out of nowhere but always take time to appear, and in each case the way an object "appears" is similar to the way some other, *"similar"*, objects appear, i.e. an object always appears as an element of a *class* of closely related objects, be it an atom, a stone, a protein, a bacterion, or a stop sign. Moreover, an object also co-evolves together with its class: there is an "invisible" permanent "bond" between an object and its class.

The crux of the proposed informational view of the universe can be expressed in the form of the *first ontological postulate*: in addition to the classes themselves, for each class there exists, in some form, (its) *class representation*, which is responsible for class integrity by guiding the generation of its new elements (and which therefore evolves together with the class). One should keep in mind that this postulate can be seen as a continuation of a very remarkable line of thought going back to Aristotle, Duns Scotus, and Francis Bacon, among others.

The *second ontological postulate* deals with the closely interconnected issue related to the nature of object representation: similar to a biological organism with its developmental "program", any object in nature also coexists with its actual formative/generative history, which must be recoverable from the above class representation. Hence the "similarity" of objects should now be understood as the "similarity" of their formative histories. Since it is the object's formative history that reveals its similarity or dissimilarity with other objects, this formative history must be captured in the object's "representation" (which is, in a sense, a generalization of the fact that all biological organisms have to store their developmental information). Actually, it is not difficult to see that the two postulates are closely related, so that one "leads" to the other.

Consistent with the above two postulates, one should approach the task of developing a formalism for structural representation as that of developing a *class-oriented representational formalism* [6], which should automatically ensure that the formalism will be suitable for the purposes of inductive learning—since the object representation would carry much more information about its class representation—and consequently for the purposes of AI in general.

It is useful to emphasize that the development of such an informational formalism should not wait for the verification of the above postulates: for one thing, no existing informational formalism insisted on any underlying hypothesis about the informational structure of the universe. I believe, however, that in contrast

[6] My papers [4], [5] are recommended as clarifying related issues.

to the latter practice, it is important to state up front an informational hypothesis which would clarify and inform the development of the corresponding informational formalism.

In what follows, I will proceed under the assumption that the above two postulates are adopted, at least as far as the development of a formalism for structural representation is concerned. Again, I would like to emphasize the importance of such adoption: without the appropriate guiding considerations, the very notion of structural object representation becomes far too ambiguous, as can be seen from its usage so far. For example, strings and graphs are not appropriate models for structural representation, since, as mentioned above, neither a string nor a graph carry within itself sufficient information for the inductive recovery of the class *from which they are supposed to come*. Mainly, this is a consequence of the situation when none of the above two postulates have been seriously considered.

One more point regarding these postulates. If we recall Georg Cantor's conception of the set as "the multitude that might be thought as oneness", we can see that *the most natural way* to conceive this multitude as one is to view the set as a class (which is specified via some generative mechanism, see section 4). In fact, I have always seen the concept of class—with its intension ("oneness") and extension—as the most natural substitute for the concept of set.

To summarize, within the proposed "computational" setting, the underlying formal structure comes from that of the class.

3 The Price of Relying on Conventional Discrete Representations

In this section, I want to address briefly the issue why the representation should be of major concern when developing a computational model. First of all, I assume (correctly or incorrectly) that the main objective of the present series of conferences is to converge on *the concept of computation* that would be viable not just within the "classical", i.e. "logical", setting but also in the natural sciences in general.

The main problem with the conventional representations, such as strings, graphs, etc., is that they do not incorporate *within themselves* any particular (generative) structure that might be associated with their formation. However, as postulated above (see the two postulates in the last section), this generative/formative information must be an integral part of an object's identity, and when absent creates an unrealistic and ambiguous situation as to this identity. Such representational deficiency (in the form of the missing representational information) is obviously *impossible to overcome* by any algorithmic means. For example, given a string **abbaca**, there is simply no way to know the sequence of operations that were responsible for the string's formation and, potentially, there are exponentially many of them. As a consequence, from the very beginning we are critically handicapped in regards to our ability to discover the underlying

generative structure, and no computational setting would be able to overcome this representational deficiency.

One might think that there is a simple way out of this situation: embed the necessary formative information (via some tags) into the representation, end of the story. However, this is a typical representational kludge, as opposed to a scientific approach: we have not gotten wiser neither about structural object representation, nor about the concept of "formative history".

So now we come to a considerably less trivial question: What is a general formalism for structural representation that would also explicate the notion of *structural generativity* as its integral part? Note that conventional computational formalisms, for example Chomsky's generative grammars, have not addressed this issue simply because they had implicitly assumed that strings are *legitimate* forms of representation, so that this more fundamental *representational issue* has not even appeared on the horizon (in spite of the fact that in this particular case Chomsky, from the very beginning, had emphasized the importance of generativity). Of course, one can excuse these *early* developments since at the time of their emergence, 1930s to 1950s, the serious representational issues, could not have appeared on the horizon; but now, at the beginning of the 21st century—when all kinds of applications, starting from general search engines (such as Google) and biological databases, all the way to various robotic applications "beg" for structural representations—the story is quite different.

Thus, the price of relying on conventional discrete representations is that the *more fundamental, underlying, representational formalism remained in the dark*, which, in turn, prevented the development of a scientifically much more satisfactory and definitive framework for computation, including the realization of the full potential of many interesting ideas such as, to take a recent example, the membrane computing.

For a discussion of the inherent, or structural limitations of the two main conventional formalisms—vector space, or numeric, and logical—see [4], [5]. Incidentally, it is these inherent limitation of the numeric (representational) formalism that, I believe, are responsible for the predominance of statistical over structural considerations in machine learning, pattern recognition, data mining, information retrieval, bioinformatics, cheminformatics, and many other applied areas. The question is not whether the *appropriate* statistical considerations should play some role in these areas, the obvious answer to which is "yes, of course", but whether, *at present*—when we lack any satisfactory formalism for structural representation—we should be focusing on the development of new statistical techniques for the conventional formalisms that are *inherently* inadequate for dealing with classes (and hence, with the class-oriented needs of the above areas).

4 A Brief Sketch of the ETS Formalism

Unhappily or happily, the structure of the ETS formalism has absolutely no analogues to compare it with, despite the fact that its main entities, "structs",

may have a superficial resemblance to some other known discrete objects such as, for example, graphs, which is useful to keep in mind. Also, in view of limited space, in what follows I restrict myself to informal descriptions, while the formal definitions can be found in [6], Parts II and III. The most important point to keep in mind is that all objects in the formalism are *viewed and represented* as (temporal) structural processes, in which the basic units are *structured events, or primitives*, each responsible for transforming the flow of several "regular" processes (see Figs. 1, 2).

4.1 Primitive Transformations

Thus, the *first basic concept* is that of a **primitive transformation**, or primitive *event*, or simply **primitive**, several of which are depicted in Fig. 1. In contrast to the basic concepts in conventional formalisms, this concept is relatively non-trivial, *carrying identical semantic and syntactic loads*. It stands for a fixed kind of *"micro-event"*, or interaction, responsible for transforming one set of adjacent processes, called *initial processes*, into another set of processes, called *terminal processes* (in Fig. 2, both are shown as lines connecting primitives). In other words, the concept of primitive transformation encapsulates that of a "standard" interaction [7] of several processes, where each belongs to a particular class of primal processes, or a **primal class** [8] . One can assume that the processes involved are "periodic" and the event is responsible for their partial or complete modification into another set of periodic processes. The formal structure of the event is such that it does not depend on the concrete initial (or concrete terminal) process, as long as each of the processes involved belongs to the same (fixed) primal class of processes depicted in Fig. 1 as a small solid shape at the top (or at the bottom) of a larger shape denoting event. As one can see, *at this, basic (or* 0^{th}*), stage of representation* [9], the structure of the initial and terminal processes is suppressed, as is the *internal* structure of the transforming event itself, and what's being captured by the formal structure is the "external" structure of the event.

Since all of nature is composed of various temporal processes, examples of the above events are all around us: e.g. an elementary particle collision, formation of a two-cell blastula from a single cell (initial process is the original cell and the terminal processes are the resulting two cells), collision of two cars, the event associated with the transforming effect on the listener's memory of the sentence "Alice and Bob had a baby" (initial processes are Alice and Bob and the terminal processes are Alice, Bob, and the baby), etc., where each mentioned event transforms the "flow" of the corresponding stable/regular processes involved.

---

[7] The *internal* structure of such event/interaction is undisclosed.

[8] The concept of class permeates all levels of consideration.

[9] In this paper, I discuss almost exclusively a single-stage version of ETS. For *multi-stage* version see [6], Part IV.

4.2 Structs

The *second basic ETS concept* is that of a **struct** formed by a (temporal) sequence of the above primitive events, as shown in Fig. 2. It is easy to see how the temporal process of Peano construction of natural numbers (Fig. 3) was generalized to the construction of structs: the single "structureless" unit out of which a number is built was replaced by multiple structural ones, i.e. by ETS primitives. An immediate and important consequence of the multiplicity of units in the construction process is that we can now see *which unit* was attached and *when*. Hence, the resulting (object) representation for the first time embodies both temporal and structural information in the form of a formative, or generative, object history recorded as a series of (structured) events. This was one of the basic driving motivations for the development of ETS, while the other main motivation was the development of a unified *class-oriented* representational formalism, which would satisfy the needs of a large variety of information-processing areas as well as natural sciences.

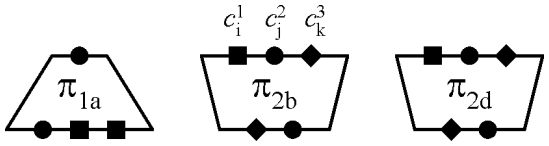

Fig. 1. Pictorial illustration of three primitives. The first subscript of a primitive stands for the event type, i.e. it refers to the set of primitives all sharing the same structure, e.g. π_{2b} and π_{2d}. The initial *classes* are marked as various solid shapes on the top, while the terminal *classes* are shown as solid shapes on the bottom: each shape stands for a particular class of processes. The only concrete processes—i.e. elements of these classes—are identified in the figure as the initial processes of primitive π_{2b} with the second subscript $\mathrm{b} = \langle\, c^1_i\,,\, c^2_j\,,\, c^3_k\,\rangle$, where c^s_t is the t^{th} initial process from the primal class C_s, $s = 1, 2, 3$.

Before proceeding further, it is important to note that there are two natural contexts within which the above concept of an object's formative history can appear. *On the agent's side*, a struct is a recorded sequence of sensory micro-events as they occur *during the agent's sensory interaction with the target object*, which must rely, of course, on the agent's own arsenal of primitives. *On the more "objective" (agent independent) side*, a struct is the representation of the sequence of *events that actually participated in the object's formation*, i.e. in the object's evolution. The essential point to observe is that both modes of representation are captured within the same formalism, which is a very desirable feature of a representational formalism, since such state of affairs suggests that the biological form of information processing didn't have to be "invented" from scratch.

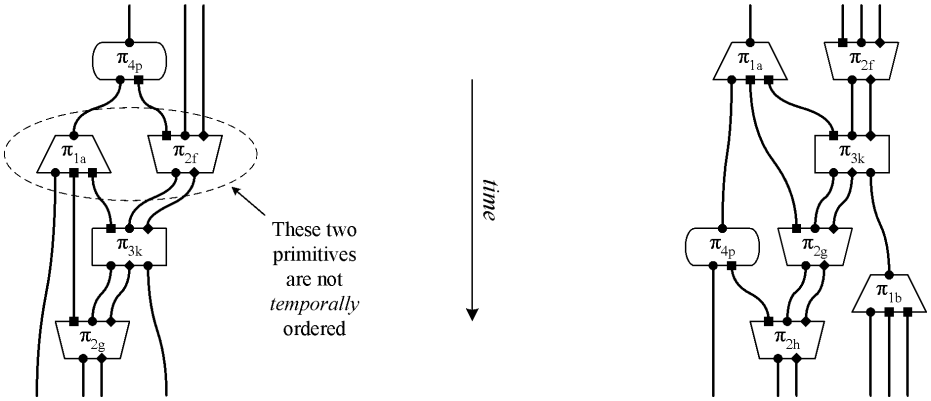

Fig. 2. Two illustrative examples of (short) structs.

Fig. 3. The single primitive involved in the ETS representation of natural numbers (left), and three structs representing the numbers 1, 2, and 3.

4.3 Struct Assembly

One of the basic *operations on structs* is **struct assembly** (illustrated in Fig. 4), which relies on the events shared by the structs involved. This operation allows one to combine in one struct several *separately observed*, but typically overlapping, structs (e.g. those representing several facial features such as the eyes and the nose). In particular, *the overlap* of several structs captures a "non-interfering", or non-destructive, *interaction* of the objects/processes that the structs represent. I should note that, in light of ETS, the noun "process" captures the conventional concept of "object" much more adequately, simply because an object is, in fact, a temporal concept.

It is not difficult to see the fundamental difference between a struct and, for example, a string: the main difference has to do with the temporal nature of ETS representation, which now allows one to "compare" any objects *based on their "formative history"*. The latter information is simply *not available in all conventional forms of representation.*

Thus, to repeat, the ETS object representation captures the object's formative/generative history which, for the first time, brings considerable additional

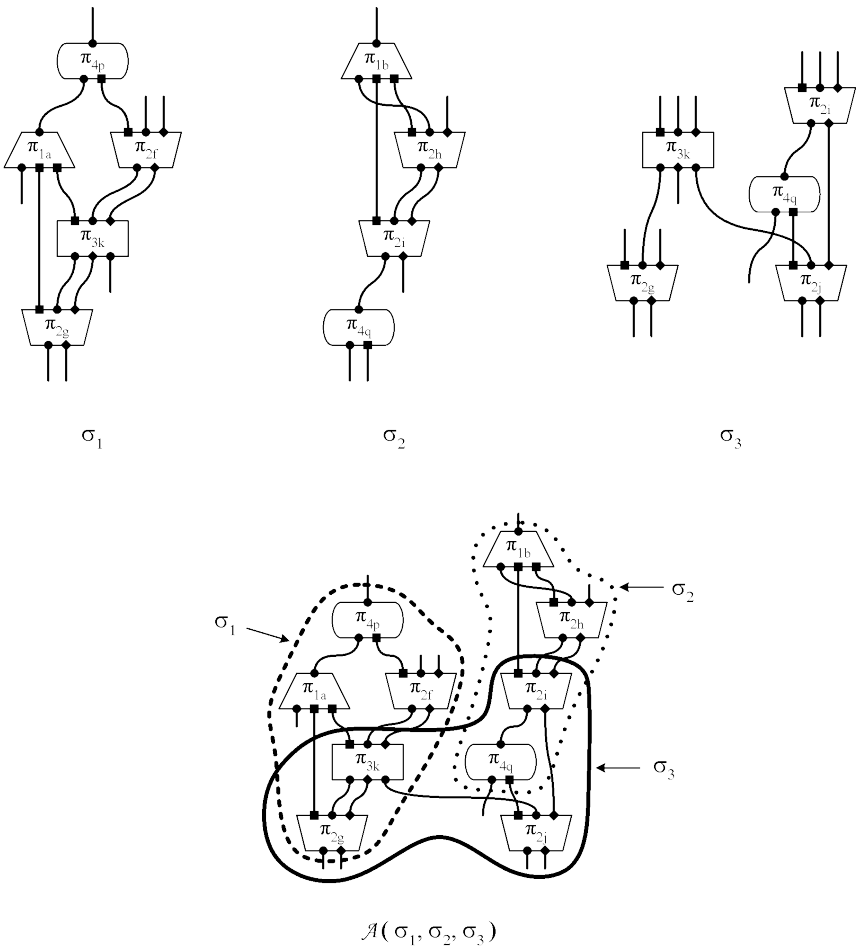

Fig. 4. Pictorial illustration of three structs and their assembly (bottom). Note that the second link connecting primitives π_{3k} and π_{2g} in the assembly comes from σ_1 (and not from σ_3).

information into the object representation. I should add that, *from the applied point of view*, the adjective "formative" does not (and cannot) refer to the object's *actual* formative history, since, in most cases, it is not accessible to us, but rather to a particularly chosen application mode of approaching/recording it, i.e. to the mode of struct construction selected by the application developers.

4.4 Level 0 Classes

The *third basic concept* is that of a **class**. Even within the basic representational stage—the only one I discuss here—each class [10] is also viewed as possibly multi-leveled. A *single-level, or 1-level, class representation* is specified by means of a **single-level, or level 0, class generating system**, which details the stepwise mode of construction of the class elements. [11]

Each (non-deterministic) step in such a system is specified by a set of (level 0) *constraints*. Each constraint—also a major concept, which is not introduced here, see [6], Part III—is a *formal* specification of a family of structs sharing some structural "components" in the form of similar substructs. During a step in the class element construction process, the struct that is being attached (at this step) to the part of the class element that has been assembled so far must satisfy one of the constraints specified for this step. To be more accurate, in the definition, it is assumed that each such step can be preceded by a step executed by the "environment", i.e. by some other class generating system "intervening" or "participating" in the construction process (see Fig. 5). Thus, quite appropriately and realistically, a change in the environment (i.e. in some of its classes) may change the class elements, without an attendant change in the class generating system itself. Such a concept of class admits the effects of the environment in a "natural" manner.

4.5 Level 1 Structs

Suppose that an agent has already learned several level 0 classes, which together form the current *level 0 class setting*. Then, when representing objects, the agent now has an access to a more refined form of object representation than a plain level 0 struct: it can now see if this struct is in fact *composed* of several level 0 class elements (each belonging to one of the classes in the level 0 class setting, see Fig. 6). This leads to the concept of the next level (level 1) struct, which provides extra representational information as compared to the underlying level 0 struct in the form of the appropriate partition of the latter struct.

4.6 Higher Level Classes

In the (recursive) *k-level* version of the *class representation*, for $k \geq 2$, each step is associated with a set of *level* $(k-1)$ *constraints*. However, during the corresponding construction process, level $(k-1)$ struct that is being attached

[10] Here, I do not include the primal classes, which are assumed to be of undisclosed structure.

[11] Note the alternative terminology: "single", or "1", refer to the *number of levels*, while "level 0" refers to the *name of the level*. Also note that since we begin with level 0, this level cannot recurse to any lover level; level 1 is built on top of level 0; level 2 is built on top of level 1; etc. Also note the difference between a *level* and a *stage* (see section 4.7): levels refer to those of classes and appear within a single representational stage.

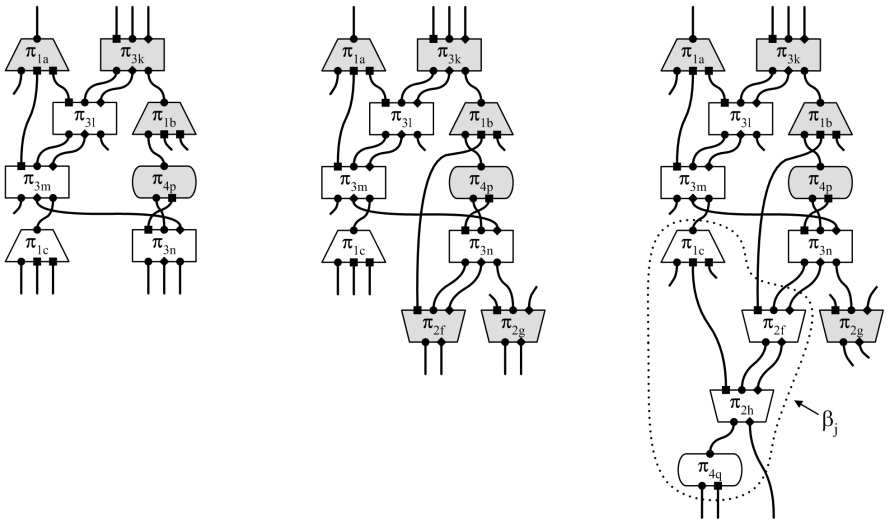

Fig. 5. Pictorial representation of a generic two-step "unit" in the construction of a class element: a step by the environment (bottom shaded primitives in the second struct, which was added to the first one) followed by a step made by the class generating system (substruct β_j in the third struct, which was attached to the second struct).

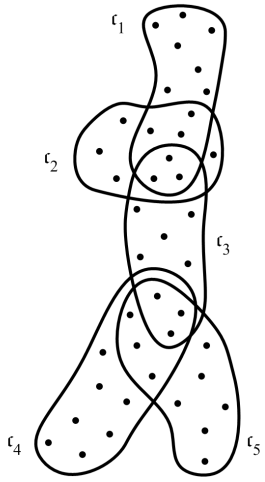

Fig. 6. Simplified pictorial representation of a level 1 struct in the contracted form: dots stand for primitives and solid lines delineate level 0 class elements $\mathbf{c}_i$'s.

(at this step) to the previously constructed part of the class element must now

be composed only out of level $(k-2)$ admissible class elements [12] in a manner satisfying one of the constraints specified for this step.

Figure 7 illustrates a (constructive) unit of such a construction process for a level 1 class element. For a level 2 class element, this element is an output of a level 2 (or three-levels) class generating system, and at each step of its construction, the relevant part is assembled out of several level 1 class elements in accordance with one of the constraints specified for this step.

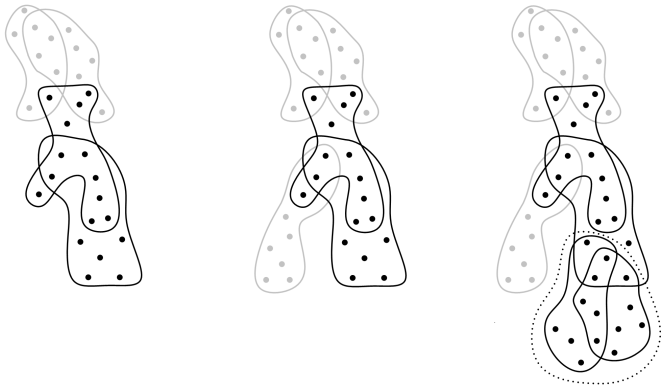

Fig. 7. Illustration of a generic two-step unit in the construction of *level 1* class element. Dots stand for primitives, and solid lines delineate level 0 class elements, which now serve as basic constructive units. A step by the environment (bottom left gray addition in the second struct) is followed by a step made by a level 1 class generating system itself (a substruct delineated by the dotted line).

4.7 Transition to the Next Representational Stage

In this section, I outline very briefly the concept of the next stage of representation, mainly because it clarifies the nature of the very basic ETS concept, that of a primitive.

A transition to the next stage of representation is associated with a representational change, more accurately compression, in which certain recurring (global) patterns of *process interactions*, called transformations, are compressed into new primitive transformations (for the next stage): each of the interacting processes is compressed into a primal process and the segment in which the interaction between them occurs is compressed into a next stage primitive event (see Fig. 8).

Thus, the primitives at the next representational stage are transformations from the present stage, including possibly some primitives from the present stage.

[12] Each of those must come from a class belonging to a (previously learned or given) set of level $(k-2)$ classes, comprising *level* $(k-2)$ *class setting.*

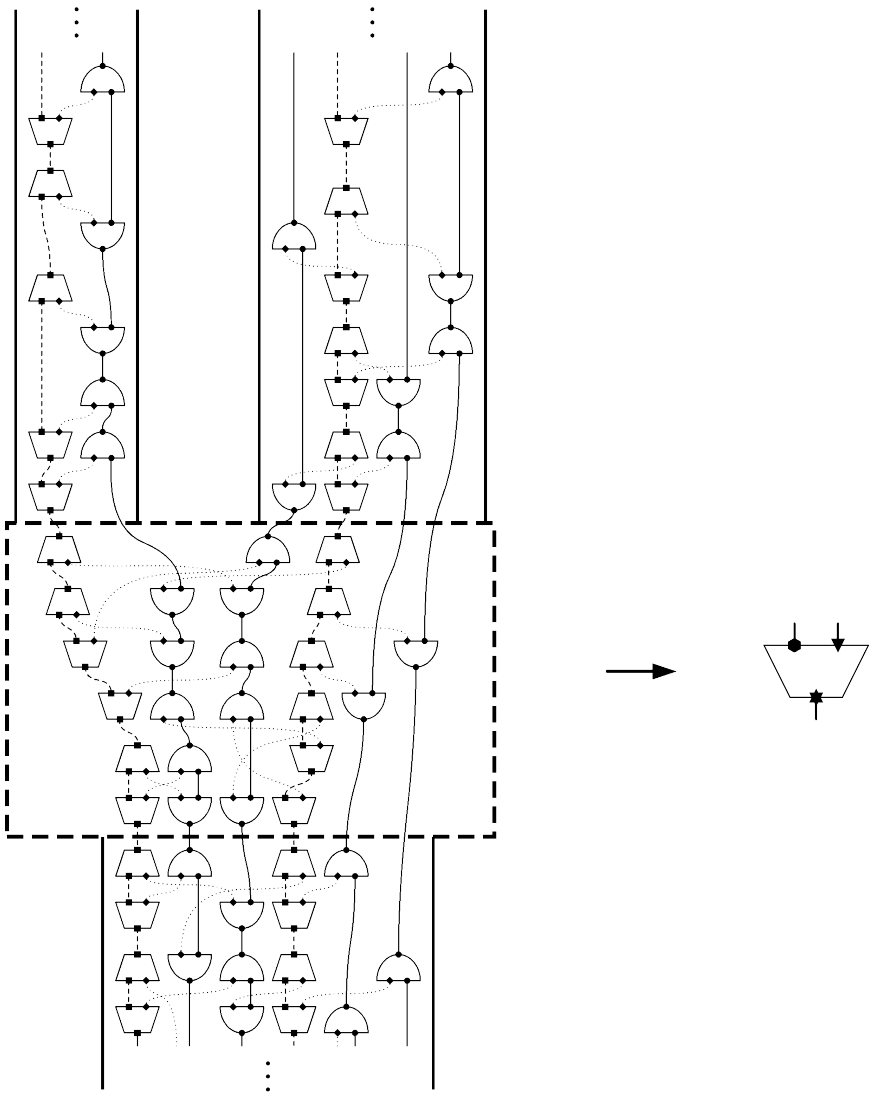

Fig. 8. A transformation (left) and the corresponding next-stage primitive (right). An illustration of a (possible) transformation corresponding to a hypothetical formation of a lithium hydride molecule (terminal process) from hydrogen (left) and lithium (right) initial processes (note the reoccurring structural patterns). The four primitives represent emission/absorption of a photon by electron (semi-circle)/nucleus (trapezoid). The body of the transform (heavy dashed line) depicts an imaginary restructuring of the two initial processes into the terminal one. On the right, the corresponding *next-stage* primitive (lithium hydride formation) is shown.

Obviously, after such compression the resulting, next stage, representation, is considerably simplified. Moreover, the transition to the next stage is accomplished seamlessly, within the confines of the same formal language.

5 Some Initial Computational Questions Arising within the ETS formalism

For simplicity, I discuss several issues and only as they relate to a *basic, single-stage*, computational "agenda" within the ETS formalism. The central problem can be stated as follows: *Given a (particular) family of multi-level classes, i.e. of their class generating systems, and given several 0-level ("training") structs from a class belonging to this family, the goal is to construct the "description" of the latter class, i.e. its class generating system.*

In other words, I suggest that within such (inductive) framework, the computational theory is aimed at developing the theory and techniques that would allow one, *first*, to preprocess (or partition, or "mark up") the input structs in a consistent manner, based on the structure of the given family of multi-level classes; and only then to proceed with the extraction of the underlying class generating system.

So first of all, one needs to develop an appropriate theoretical framework, which could be thought of as a "structural" generalization of the Chomsky's language hierarchy. As mentioned in the first paragraph of this section, such a theory would allow one to properly restrict the learning problem to a particular family of classes. The development of such a hierarchy should probably proceed along the lines somewhat similar to those followed by Chomsky, with the following quite natural adjustments. First, some levels in the hierarchy might need (internal) partitioning; and second, within the hierarchy, each family of classes should be introduced via the appropriate restrictions on the set of admissible constraints—which are the ETS analogues of the production rules—involved in the specification of the class. However, it is important to emphasize the differences between the ETS and the conventional string (and graph) hierarchies, which concern not just the presence of levels in ETS. When defining a conventional, for example Chomsky, hierarchy, *one is not treating a string as a form of structural object representation* as understood above. This results, on the one hand, in a wider range of admissible production rules (not all are meaningful from the representational point of view), and on the other hand, in a variety of qualitatively different machines associated with each family of languages (e.g. deterministic/nondeterministic). There are reasons to believe that, in the case of ETS formalism, in view of the temporal form of representation, the situation is more palatable.

Next, based on such a structure theory, one would need to develop techniques for the above *class related* (structural) preprocessing, or partitioning, of the structs in a given (training) set of structs. [13] Even this task is highly non-

[13] Although I'm not discussing this issue here, in many applications, one would need to do all preprocessing under the assumption that some cyclic permutations of structs

trivial, since the tiling of each given struct must be accomplished relying on the *admissible* previous level class elements only (see Fig. 6). The final stage is related to the construction of the corresponding class representation.

It is understood that, in the meantime, to proceed with various applications, one does not need to wait for the development of a full-fledged structural theory, since in each concrete application, the corresponding family of classes can readily be specified.

Thus, it should be clear that in contrast to the conventional computational models—where, as was mentioned above, the defining agenda was logical—I propose that it is the (structural) inductive agenda that should now drive the development of the computational framework. This proposal is not really surprising, given that the new framework is aimed at supporting a continuous dynamic interaction between the "machine" and various *natural environments*, where, as was hypothesized above, the temporal/structural nature of object representation is ubiquitous.

6 Conclusion

I hinted that it might be prudent for some researchers in computation to add to their research interests a new area that we called *Inductive Informatics*, which can be thought of as a reincarnation of a four-hundred-year-old Francis Bacon's project of putting all sciences on firm inductive footing, with the attendant restructuring of the basic formal/scientific language and methodology. Interestingly, the future of what we now call "computer science" appears to lie in this direction. In hindsight, it looks as if the major obstacle to Bacon's vision has been the lack of a classification-oriented—or which appears to be the same thing "structural"—representational formalism. Our proposed version of such formalism is a far-reaching, structural, generalization of the numeric representation, and in that sense *it is much closer to mathematics than the logic* (which was suggested by von Neumann [2] to be a desirable direction). Moreover, with the appearance of such a formalism, the concept of *class representation*, for the first time, becomes meaningful, and it also becomes quite clear that it is the *intrinsic* incapability of the conventional representational formalisms to support this fundamental concept that is responsible for our previous failure to realize Bacon's vision.

In addition to the new theoretical horizons that are being opened up within new (event-based) representational formalisms, the other main reason why such formalisms should be of interest to researchers in computation has to do with the *immediate* practical benefits one can derive from their various applications in machine learning, pattern recognition, data mining, information retrieval, bioinformatics, cheminformatics, and many other applied areas. The legitimacy of the last statement should become apparent if one followed carefully the above

and/or some of their substructs should be treated as identical representations by the agent, since they may correspond to alternative ways of sensing the same object in the environment.

points regarding the new and rich (temporal) aspect of object representation—i.e. the object's formative history—as well as the (generative) concept of class representation that now become available within the ETS formalism.

Finally, it is useful to note the expansion of event-based research directions in recent (last 30-40 years) work in philosophy, psychology, and linguistics (e.g. [7], [8], [9], [10], [11]).

Acknowledgment. I thank Oleg Golubitsky for a discussion of the paper and Ian Scrimger and Reuben Peter-Paul for help with formatting.

References

1. Cook, S. A.: The complexity of theorem proving. In: Laplante, P. (ed.): Great Papers in Computer Science. West Publishing Co., St. Paul, Minneapolis (1996) 2
2. Neumann, von J.: The general and log ical theory of automata. In: Pylyshyn, Z. W. (ed.): Perspectives in the Computer Revolution. Prentice-Hall, Englewood Cliffs, New Jersey (1970) 99–100
3. Turing, A. M.: On computable numbers, with an application to the Entscheidungsproblem. In: Laplante, P. (ed.): Great Papers in Computer Science. West Publishing Co., St. Paul, Minneapolis (1996) 303, 304
4. Goldfarb, L.: Representational formalisms: What they are and why we haven't had any. In: Goldfarb, L. (ed.): What Is a Structural Representation (in preparation) http://www.cs.unb.ca/~goldfarb/ETSbook/ReprFormalisms.pdf
5. Goldfarb, L.: On the Concept of Class and Its Role in the Future of Machine Learning. In: Goldfarb, L. (ed.): What Is a Structural Representation (in preparation) http://www.cs.unb.ca/~goldfarb/ETSbook/Class.pdf
6. Goldfarb, L., Gay, D., Golubitsky, O., Korkin, D.: What is a structural representation? A proposal for an event-based representational formalism. In: Goldfarb, L. (ed.): What Is a Structural Representation (in preparation) http://www.cs.unb.ca/~goldfarb/ETSbook/ETS6.pdf
7. Casati, R. and Varzi, A.: Fifty Years of Events: An Annotated Bibliography 1947 to 1997, Bowling Green (OH), Philosophy Documentation Center (1997) (also http://www.pdcnet.org/eventsbib.htm)
8. Casati, R.: Events, Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/entries/events (2006)
9. Lombard, L. B.: Events: a Metaphysical Study, Routledge and Kegan Paul, London (1986)
10. Zacks, J. M. and Tversky, B.: Event structure in perception and conception, Psychological Bulletin **127** (2001) 3–21 (also http://www-psych.stanford.edu/~bt/events/papers/eventspsychbull.pdf)
11. Tenny, C, and Pustejovsky, J. (ed.): Events as Grammatical Objects, CSLI Publications, Stanford, CA (2000)