MULTI-HIERARCHICAL LEARNING-BASED COSIMULATION

TUDOR NICULIU
«Politehnica» University Bucuresti, Faculty of Electronics
Splaiul Independentei 313, 77206 Bucuresti, Romania
tudor@messnet.pub.ro

SORIN COTOFANA
Delft University of Technology, EE Faculty
Mekelweg 4, 2600 GA Delft, The Netherlands
S.D.Cotofana@dutepp0.et.tudelft.nl

ABSTRACT

Intelligence supposes, at least, consciousness and adaptability. Consciousness simulation demands transcending the present limits of computability, by an intensive as well as extensive research effort to integrate essential physical and mathematical knowledge guided by philosophical goals. Separating the different hierarchy types reveals their comprehensive constructive importance based on structural approach, symbolic meaning, object-oriented representation, their combination in looking for self-organization, self-control and conscience. Multiple, coexistent and interdependent hierarchies structure the universe of models for complex systems, e.g., hardware/ software ones. They belong to different hierarchy types, defined by abstraction levels, block structures, classes, symbolization and knowledge abstractions. Abstraction and hierarchy are semantic and syntactical aspects of a unique fundamental concept, the most powerful tool in systematic knowledge; hierarchy results by formalizing abstraction; only, hierarchies are still constraint by computability, while abstraction assists beyond learning also consciousness and intuition. Hierarchies of different types correspond to the kind of abstraction they reflect. The gap that appears between reflexive abstraction and hierarchical computation is an important challenge for the mathematical and scientific community.

KEYWORDS: Hierarchical Intelligent Simulation, Hierarchy Types, Cosimulation.

INTRODUCTION

Artificial Intelligence means simulation of intelligence, either behavioral (functional or procedural) or structural (e.g., neural, genetic, cellular). The first way should not be neglected, although the second one is actually more efficient for bottom-up learning. Only hierarchical simulation, assisted mathematically to build theories and formalisms, can lead to understand the results, so to manage them truly. Hierarchical approach should concentrate on knowledge hierarchies, to enable metaknowledge simulation, for the system's adaptability, but also for looking for the way to simulate consciousness as good as possible. We present the different hierarchy types, respective formalization ideas and examples, emphasizing the knowledge hierarchy types. The last section presents a hierarchical schema for hardware/ software cosimulation. Conclusions and important references end the paper.

SIMULATED INTELLIGENCE

Intelligence supposes, at least, consciousness and adaptability. Consciousness simulation demands transcending the present limits of computability, by an intensive as well as extensive research effort to integrate essential physical and mathematical knowledge guided by philosophical goals. Separating the different hierarchy types reveals their comprehensive constructive importance based on structural approach, symbolic meaning, object-oriented representation, their combination in looking for self-organization, self-control and conscience. Knowledge and construction hierarchies can cooperate to integrate design and verification into simulation; object-oriented concepts can be symbolized to handle data and operations formally; structural representation of behavior manages its realization. Hierarchy types open, or at least show, the way to simulate intelligence as adaptable consciousness, offering a third alternative way to look for extending the limits of computability; this should be integrated with the first two: heuristic - risking correctness for completeness, and scientific - imitating biochemical structures or looking for a quantum revolution; the three choices remind of Thomas Mann's Zauberberg, Johann Wofgang von Goethe's Faust and Hermann Hesse's Glasperlenspiel.

HIERARCHY TYPES

Generally, multiple, coexistent and interdependent hierarchies structure the universe of models for complex systems, e.g., hardware/ software ones. They belong to different hierarchy types, defined by abstraction levels, block structures, classes, symbolization and knowledge

abstractions. Abstraction and hierarchy are semantic and syntactical aspects of a unique fundamental concept, the most powerful tool in systematic knowledge; hierarchy results by formalizing abstraction. Hierarchies of different types correspond to the kind of abstraction they reflect (abstraction goal is shown between brackets):

Class hierarchy (↑concepts) ↔ virtual framework to represent any kind of hierarchy, based on form-contents dichotomy (class-instance), modularity, inheritance, polymorphism; an object is defined by identity, state and behavior, being instance of a class, that defines its structure and behavior (internal - completing the structure, external - for communication); to exist behaviorally, the object hides its structure, thus integrating in a world of adaptable objects that intercommunicate, developing towards conscious and intelligent objects, i.e. subjects.

Symbolization hierarchy (↑mathematics) ↔ stepwise formalism for all types, e.g., for hierarchy types.

Structure hierarchy (↑managing) ↔ stepwise managing of all (other hierarchy) types on different levels by recursive autonomous block decomposition, following the principle "Divide et Impera et Intellige".

Construction hierarchy (↑simulation) ↔ design/ verification (= simulation) framework of autonomous levels for different abstraction grades of description; time is explicit at highest (behavioral) levels, being integrated in the model, and exterior on lowest (structural) levels, being implicit for the system's activity; artificial intelligence approaches try to configure the simulation hierarchies as reciprocal to knowledge hierarchies.

Knowledge hierarchy (↑theories) ← reflexive abstraction ("in a deeper sense"); it aims that each level has knowledge of its inferior levels, including itself; recurrence of structures and operations enables approximate self-knowledge (with improved precision on the higher levels of knowledge hierarchies); a continuous model for hierarchy levels, without loosing the hierarchy attributes, would offer a better model for conscience and intelligence; a possible interpretation of such hierarchies is: real time of the bottom levels (primary knowledge/ behavior/ methods) is managed at upper levels (concrete types/ strategies/ models) and abstracted on highest levels (abstract types/ theories/ techniques).

The formalization of hierarchy types is guided by their relations and can be done using the theory of categories [1]; constructive type theory permits formal specification and formal verification by generating an object satisfying the specification:

- knowledge ← structure, classes, symbolization (abstraction, recurrence → reaction);
- construction ← structure, classes, symbolization (recurrence, space ↔ time);
- classification ← existential abstract types (syntax of construction/ knowledge);
- symbolization ← universal abstract types (semantics of construction/ knowledge);
- structure ← comprehensive concrete types (pragmatics of construction/ knowledge).

Understanding and construction have correspondent hierarchy types: their syntax relies on classes, their meaning on symbols, their use on modules.

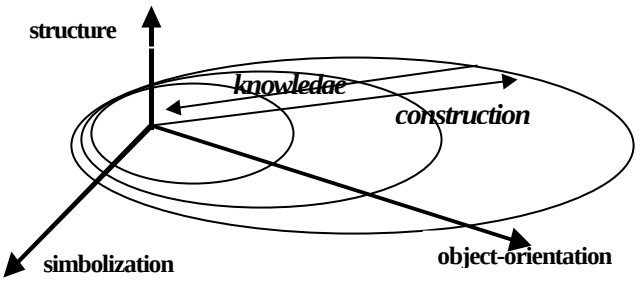

Figure 1: H - diagram

ABSTRACTION LEVELS

A net of abstraction levels, representing a unique object/ process, is defined by: $(\{L_i\}, \{A_i\})$; the abstraction relations $\{A_i\}$ induce a partial ordering on the level set $\{L_i\}$. The behavior defined on a lower level includes the behavior defined by an upper level. Abstraction represents either specialization neglecting irrelevant information (examples → prototypes), or passing from imprecise to categorical models (reality → examples).

Qualitative simulation [2] operates on the prototype level, built upon abstraction morphism (or on a collection of independent example sets).

(D, rel-abs) - *prototype* domain, structured by rel-abs;
$(E = D_1 \times ... \times D_n$, rel) - *examples* domain, rel-structured;
$A: E \rightarrow D$ (abstraction);
$K: D \rightarrow E$ (concretization);
compatibility: $A \circ K = Id_D$;
specialization: $K \circ A \neq Id_E$;
$X \subset E$, *independent set*: rel defined by representatives;
$A: E \rightarrow D$, *morphism*: transports the structure on E to D.

Figure 2: Qualitative Simulation

Qualitative simulation (qualSim) should be:

- complete, i.e., any real simulation can be represented as $K \circ qualSim \circ A$;

- correct, i.e., for any qualSim, there is K so that
  K ∘ qualSim ∘ A is a "real" simulation.

Uncertain simulation is based on a mathematical model for the transmission of uncertainty when applying simulation operations [3]: uncertainty propagation when composing operations, hypothesis combination, operation correlation. Statistical concepts model objective lack of information; fuzzy concepts describe subjective lack of knowledge.

```
Composionality:
    behavior ⇐ structure  component behavior
Causality:
    local behavior ⇐ classification ( implicit description)
Stationarity:
    behavior discretization ⇐ adequate symbolization
Functionality:
    knowledge hierarchy levels:
        - function of the system
        - procedural behavior of the system
```

Figure 3: Construction hierarchy relations to other types

BLOCK DECOMPOSITION

Structural partition is represented as a tree or a non-cyclic directed graph; it is approached by the "Divide et Impera et Intellige" principle:

```
template <class ProblemType, class SolutionType>
  class StructureHiearchy {
        LIST <ProblemType> subproblems;
        LIST <ProblemType> solutions;
        BOOLEAN simple (<ProblemType>);
        <SolutionType> solution (<ProblemType>);
        LIST <ProblemType> decompose (
                                        <ProblemType>);
        <SolutionType> compose (LIST <SolutionType>);
        LIST <SolutionType> add(SolutionType>);
    public:
        <SolutionType> DivImpInt (
                                <ProblemType> problem) {
                IF simple (problem)
                        RETURN (solution (problem));
                subproblems ⇐ decompose (problem);
                FOR EACH problem IN subproblems
                    solutions ⇐ add (DivImpInt (problem));
                RETURN (compose (solutions));
                }
        };
```

Figure 4: Divide et Impera et Intellige

CONCEPT HIERARCHY

The hierarchical principle applies to the approach (a problem is solved as an effect of representing its domain and some stimuli), as well as to the structure of knowledge (it mediates the action of a paradigm on an environment). Knowledge-based architecture, both at environment and simulation component level, ensures flexibility of the framework realization, by defining it precisely only in the neighborhood of solved cases [4].

```
OBJ = interpretation (obj1, ..., objm);
(obj1, ..., objm) = implementation (obj'1, ..., obj'm');
    OBJ - abstract object;
    obj - hard/ soft object;
    obj' - concrete object;
abstraction =  interpretation o implementation:
        behavioral (functional/ procedural)/ structural;
simulation-component = (    abstraction,
                            implementation,
                            interpretation):
        module - energy/ information processing,
        connection - energy/ information transmission.
```

Figure 5: Object-oriented abstraction

SYMBOLIZATION DEGREE

Class-instance hierarchy can support symbolic operations (λ-calculus, extended to cope with program termination, i.e., partial functions) and evaluation [5]. A symbolic expression is knowledge about an expression obtained by evaluation of some symbols (metaknowledge). For linear and weakly nonlinear systems, mathematics offers symbolic methods, that directly determine the functional behavior; their instantiation result in the correspondent numerical methods. Relaxation methods result by expanding symbolization hierarchy on the base level.

METAKNOWLEDGE

Knowledge is based on a morphism that maps the state-space of the object-system onto the internal representation of the simulator. An intelligent simulator can learn by generating and validating models of the object-system [6]. Therefore: representation for design and verification should be common; the algebraic structures on which the different hierarchy types are based on should be extended to topological structures; the different simulation entities should be symbolic, having attributes as: type, domain, function. A topology on the space of symbolic objects permits grouping items with common properties in classes. A dynamically object-oriented internal representation results, that can be adapted to the different hierarchy types. Topological concepts, as neighborhood and closure, can be applied in verification and optimization, for objects and classes as well.

Knowledge-based architecture separates representation from reasoning. A system capable of reflexive abstraction ("intelligent") reasons controlled by the problem specification and by solving strategies. These are derived from a higher level of knowledge, representing principles of approach, which are structured by an even higher level, containing hierarchy types. An object-oriented simulation framework permits the representation of different knowledge levels, each having concept, symbol, structure hierarchy. For representation, this principle offers the advantage of open modeling. Models are described by the user, following a general accepted paradigm (e.g., entity-architecture decomposition) that ensures syntactic correctness, leaving the meaning to be specified by user-defined semantic functions that control the simulation. For example, a module in an unfinished design can be characterized by constraints regarding its interaction to other modules; the constraints system is a model, open to be interpreted, thus implemented, in different ways that derive from adequacy criteria in a non-monotonic logic [7].

Explanation is a key concept for knowledge-based systems. It can be expressed as proof in a deductive system, whose axioms are the equations constraining component models and input signals, theorems are simulation results, inference rules represent logic & domain-specific calculus. Using constructivism in logic, e.g., intuitionistic predicate logic [8], behavior or structure of the simulated system can be extracted from the proof.

Knowledge hierarchy is not based on simplifying abstraction, as are the other hierarchy types; it explicitly represents metaknowledge (knowledge about knowledge), based on a reflexive abstraction form, that links (abstract) objects to functions defined for these objects. Interlevel relations can be interpreted as planning (top-down) and learning (bottom-up).

Learning derives a formal structure on the upper level (e.g., a static structure), from experiences on the lower level (procedures executed using resources that are not present at the upper level, e.g., time). It has two complementary aspects: 1) induction - extensive knowledge at the lower level is transformed into intensive knowledge at the upper level, using non-reflexive abstraction (equivalence, isolation, emphasis, stationary approximation, idealization); 2) deduction - intralevel concept production (conditioning, association, stress, imitation).

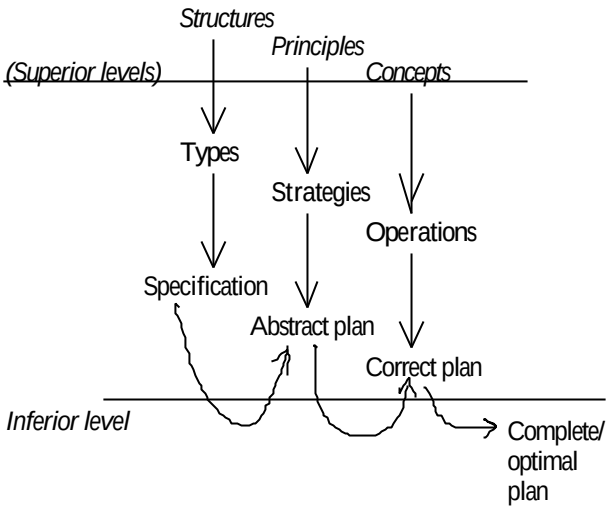

Figure 5bis: Knowledge hierarchy

Planning transforms declarative knowledge (formal, but limited) in partially procedural knowledge (unlimited, but implying a context with resources that are not formalized at the upper level; the main resource is time). Artificial intelligence studies planning as reasoning about actions; actions are elements of a lower level, generally represented by states (instances of upper level functions in the presence of a context) and operations (determining state transitions). The plan is a non-commutative system of declarative knowledge; extreme cases are: commutative rule set, sequential procedure.

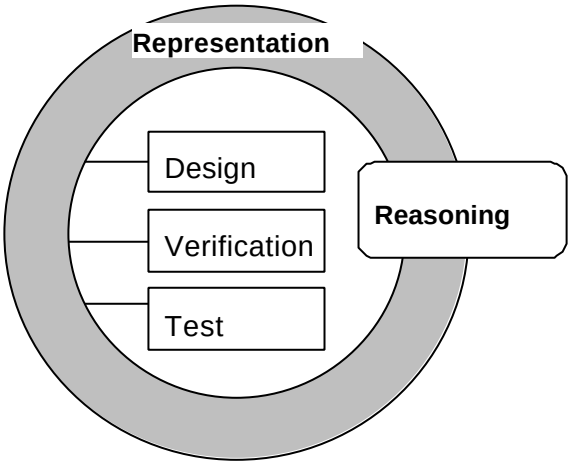

Figure 6: Knowledge-based Simulation System

TYPE THEORIES FOR SIMULATION

The constructive theory of intelligent simulation specifies its syntax and axioms its semantics based on a knowledge hierarchy: a bottom level consists in the logical base, the upper level contains the operations, and the highest level is a theory of types. Axiomatic semantics does not refer to a particular implementation of the functions, offering a

mathematical theory of intelligent simulation, where the design can be demonstrated correct regarding the specification or the equivalence of two designs can be shown. Reasoning about functions and types is performed in a predicate calculus; the intuitionistic one sustains the correct design by a constructive prove of the specification. The theory of operations gives axioms for equality, definition and application of functional operations. Termination of programs/ activities can be formalized extending general functions to partial ones. The λ-calculus offers a formal framework for these aspects. Types in simulation are important as theoretical guides and as practical aids.

HIERARCHICAL CO-SIMULATION

Intelligent systems call for hierarchical co-simulation of its hardware/ software [9], digital/ analog, thermal/ electrical etc parts, in the context of a unified representation of design and verification. Simulation (design/ verification) should remain correct, with increasing complexity and optimization requirements of the object-system. The hierarchical principle, applied to knowledge and simulation, (locally) bounds the complexity, by problem decomposition, and assures (almost) correct-by-construction designs and design-adapted verification; its apparent lack of efficiency is due to nonhierarchical optimization techniques.

The designed framework permits self-organizing, offering at any level of abstraction of the simulation hierarchy:

description of the system in a convenient language, e.g., C++ extended for parallelism by synchronization constructs; automatic learning-based partition of the description into hard/ soft; correct and complete communication between heterogeneous parts and with the exterior; simulation and validation of the whole system during any design phase.

The simulation environment prepares a framework for representing entities and relations of the system to be simulated, as well as general knowledge about the simulated universe. Objects are defined by properties (static attributes & dynamic methods): behavioral (response to stimuli received from the other objects); structural (part of a set/ configuration built from other objects); functional (formal properties).

If one of the imposed properties (design constraints), regarding communication, correctness or testability, is considered as not being fulfilled after applying a technique, using a model and suitable methods for measure and improvement, different strategies permit altering one of the technique/ model/ method, to repeat the process for the initial behavioral specification or the one resulted from prior (insufficient) improvement. This calls for an intelligent choice of the designer or the AI system that assists/ automates the design. The methods are recursive (iterative) to handle the different components in the behavioral specification of the system. The process continuation is controlled by measurement functions, so, generally, these must be called for each call of the improvement functions, but there are also methods demanding for a global improvement based on a prior measurement.

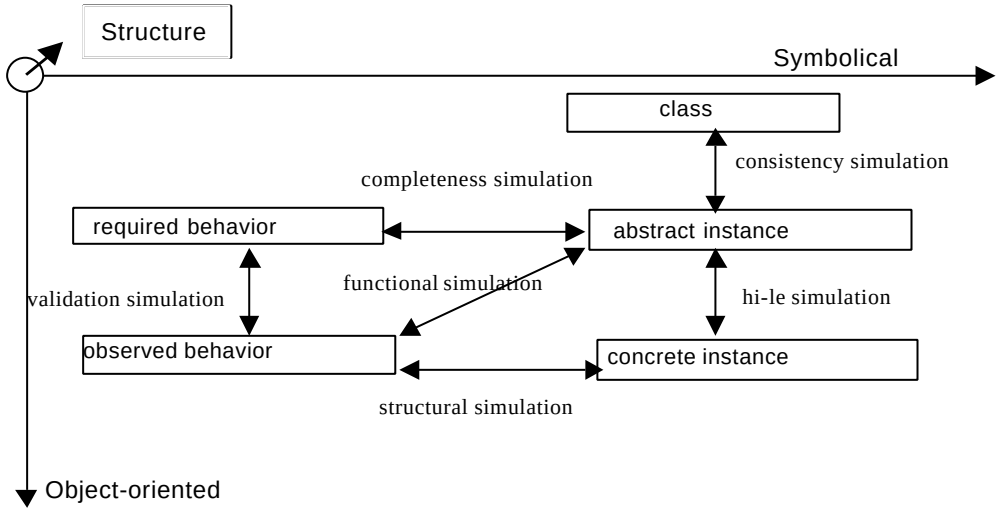

Figure 7: Hierarchical Co-Simulation Paradigm

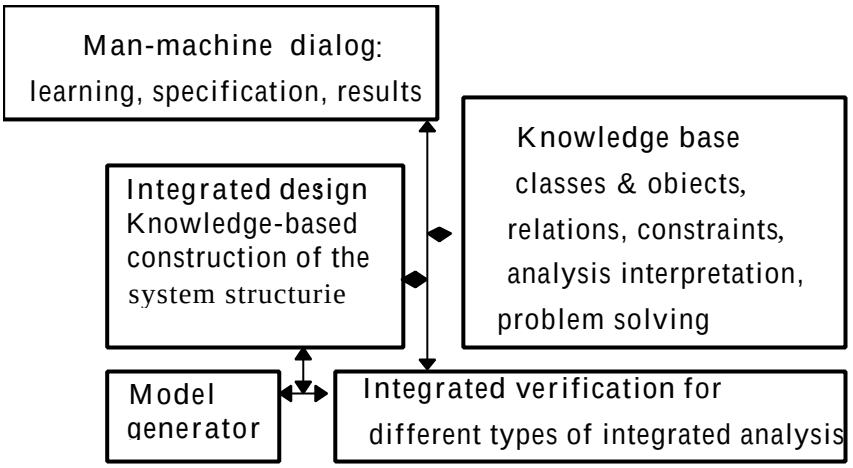

Figure 8: Behavioral Adaptable Design Framework

CONCLUSIONS

We studied hierarchy types, separately and in relation one to another, especially:

- interlevel relationships, with qualitative theories and uncertainty formalisms;
- "Divide et Impera et Intellige" (recursive approach) to solve algebraic/ differential non/linear systems, represented with sparse matrices, at different structural levels;
- object-oriented methodology, reducing multiple inheritance, due to coexistent hierarchies, by multiple entities and constraints;
- symbolical approach as an instantiation hierarchy (evaluation) or as knowledge hierarchy (function = role);
- hierarchical principle applied to the object of knowledge as to the knowledge structure itself, to mediate the action of a paradigm on an environment.

Simulation theory is based on algorithmic mathematics. Intelligence simulation implies hierarchical approach of different types. Knowledge hierarchies demand for an extension of the concept of algorithm. Hardware new architectures, to permit unlimited parallelism of the correspondent software, are needed. A systematic hierarchical approach permits representing all hierarchy types, leading to compact, non-redundant, easy to modify models, for systems as well as processes, sequential or parallel. Representations are formalized in constructive mathematics style, advancing from correct specification to correct representation and, further, to correct simulation. Combining complementary directions by object-orientation unifies the simulation methods for the different kind of parts. Knowledge-based object-oriented multi-hierarchical simulation procedures allow recurrence for different hierarchy types. Representation-inference dichotomy, that characterizes knowledge-based methods, enables reflexive abstraction, thus, producing new knowledge starting with problem specification, following solving strategies, structured by approach principles and hierarchy types.

REFERENCES

[1] F. Kasch and B. Pareigis, *Grundbegriffe der Mathematik* (München, Germany: Fischer Verlag, 1986).

[2] P. Winston, *Artificial Intelligence, 3rd Edition* (Reading, Ma: Addison-Wesley, 1992).

[3] B. Stroustrup, *The C++ Programming Language, 3rd Edition* (Reading, Ma: Addison-Wesley, 1997).

[4] G. Booch, *Object-oriented Analysis and Design, 2nd Edition* (Reading, Ma: Addison-Wesley, 1996).

[5] P. Jalote, *An Integrated Approach to Software Engineering*, (Berlin, Germany: Springer, 1991).

[6] B. Meyer, *Introduction to the Theory of Programming Languages* (Englewood Cliffs, NJ: Prentice-Hall, 1990).

[7] W. Bibel et al., *Wissensrepräsentation und Inferenz* (Braunschweig, Germany: Vieweg, 1993).

[8] R. Turner, *Constructive Foundations for Functional Languages*, (New York, NJ: McGraw Hill, 1991).

[9] D. Gajski et al., *Specification, and Design of Embedded Systems* (Englewood Cliffs, Prentice-Hall, 1994).