HIERARCHICAL INTELLIGENT SIMULATION

Tudor Niculiu
Bucharest University of Technology
Splaiul Independentei 313
77206 Bucuresti, Romania
tudor@messnet.pub.ro

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

Keywords
Hierarchical, Combined simulation, AI in simulation.

Abstract
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. Artificial Intelligence means simulation of intelligence, either behavioral (functional or procedural) or structural (e.g., neural, genetic, cellular). Only hierarchical simulation, assisted mathematically to build theories and formalisms, can lead to understand the results, so to manage them truly. The 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.

Introduction
Intelligence assumes, 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 and intuition guided by philosophical goals. An algorithm is a computer simulable entity, so it represents computability, bottom-up (construction, design, plan) or top-down (understanding, verification, learning). The algorithmic approach is equivalent to the formal one: If a sentence of a formal system is true, then an algorithm can confirm it. Reciprocally, for a verification algorithm of the mathematical sentences a formal system can be defined, that holds for true the sentences in the set closure of the algorithm's results towards the operations of the considered logic. Formal systems, partial-recursive functions, Turing machines, λ -calculus, are only the best-known formalisms for computation, i.e., for algorithm and computability.

Hierarchy Types
Multiple, coexistent and interdependent hierarchies structure the universe of models for complex systems. 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 formalizing abstraction. Hierarchy types correspond to the various abstraction ways ($\uparrow$abstraction goal):

Class hierarchy ($\uparrow$ concepts) $\leftrightarrow$ virtual framework to represent any kind of hierarchy, based on form-contents dichotomy, 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, what helps it to integrate in a world of adaptable objects that intercommunicate, developing towards conscious and intelligent objects, i.e. subjects.

Symbolization hierarchy ($\uparrow$ mathematics) $\leftrightarrow$ stepwise formalism for any kind of types, e.g., hierarchy types.

Structure hierarchy ($\uparrow$ problem-solving) $\leftrightarrow$ 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 ($\uparrow$ simulation) $\leftrightarrow$ 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 ($\uparrow$ theories) $\leftarrow$ reflexive abstraction ("in a deeper sense"); each level should know of its inferior levels, itself included; recurrence of structures and operations enables 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.

Understanding and construction have correspondent hierarchy types: their syntax relies on classes, their meaning on symbols, their use on modules (Figure 1).

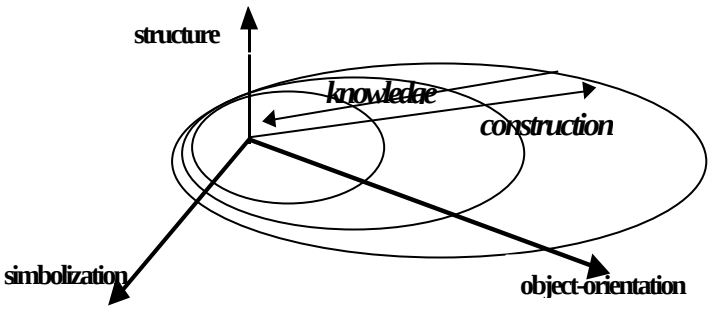

Figure 1: H - Diagram

All hierarchy types have common structures as the following:

(U, {Hi Î Sh}) - universe, with different hierarchies **Hi**,
Sh - set of hierarchies, **Sl** - set of hierarchy levels:
H = (Rel_eq, {(Levelj,Structurej): j Î Sl},
Rel_ord, {Aj: j Î Sl}) - generic hierarchy:
Rel_eq - equivalence relation, divides **U** in levels,
Structurej - structure defined of level **j**,
Rel_ord - order relation (total), defined on **Sl**,
Aj Î {(x,y):x Î Levelj-1,y Î Levelj, j Î Sl} - abstraction.

Hierarchies are leveled structures, which represent different domains. A level is an autonomous mathematical structure, containing abstract/ concrete entities, linked by intralevel relations. Abstraction relates the levels: this induces an interlevel order relation, partial, concerning entities, and total, regarding the levels. Beyond the hierarchical point of view, the system can be formalized as an autonomous domain, structured by metahierarchical relations, building a level in a higher order hierarchical system. Hierarchical structures exhibit two complementary processing strategies: top-down and bottom-up. A sketch to formalize hierarchy types follows:

- Knowledge ← structure, classes, symbolization
  (abstraction, recurrence → reaction)
- Construction ← structure, classes, symbolization
  (recurrence, space ↔ time)
- Classes ← existential abstract types
  (syntax of construction/ knowledge)
- Symbolization ← universal abstract types
  (semantics construction/ knowledge)
- Structure ← comprehensive concrete types
  (pragmatics construction/ knowledge)

The different hierarchy types can be formalized by the theory of categories (Kasch and Pareigis 1986). Constructive type theory permits formal specification as well as formal verification by generating an object satisfying the specification.

Example: The classical activities in complex systems simulation, that regard different levels of the construction or knowledge hierarchy, can be expressed symbolically then represented object-oriented and simulated structurally, as sketched in Figure 2:

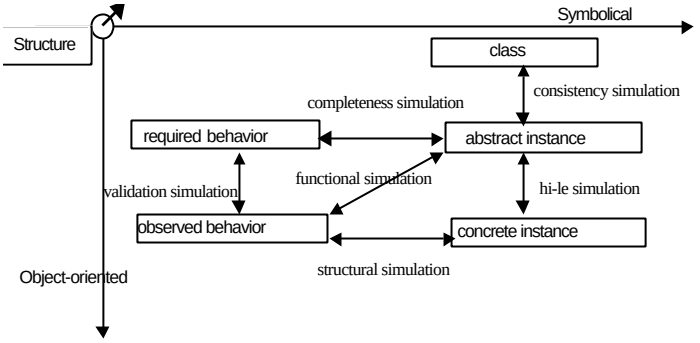

Figure 2: Hierarchical Simulation Paradigm

A simulation approach/ paradigm is interesting only when summing more of the following attributes: incremental, algorithmic, integrated, hierarchical, intelligent, inspired:

Construction ⇐ inspired, intelligent, concrete
Knowledge ⇐ integrated, intelligent, abstract
Intelligence ⇐ conscious, adaptable
Consciousness ⇐ knowledge hierarchy, incremental
Adaptability ⇐ simplifying hierarchies + algorithm
Will ⇐ | conscious - adaptable |

The simulation framework permits self-organizing, offering at any level of abstraction of the simulation hierarchy: description of the system in a convenient and commonly used language, e.g., C++ extended for parallelism by synchronization constructs; automatic learning-based partition of the description into hardware and software; correct and complete communication between heterogeneous parts and with the exterior; simulation and validation of the whole system during any design phase (Figure 3).

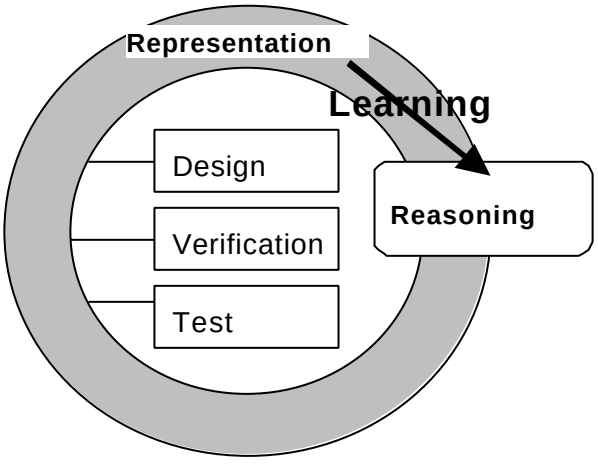

Figure 3: Knowledge-based Simulation Framework

If one of the imposed properties (design constraints) is not 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 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.

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 proof 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. λ -calculus offers a formal framework for these aspects. Types in simulation are important as theoretical guides and as practical aids.

Hierarchical to Intelligent

The planned simulation environment prepares a framework for representing entities and relations of the system to be simulated (designed/ verified), as well as general knowledge about the simulated universe. Objects are defined by properties: structural attributes (part of a set/ configuration built from other objects) and behavioral methods (response to stimuli received from the other objects). Symbolization hierarchies are used to manage formal representation of the previous kinds of properties, as well as of the knowledge abstraction.

Knowledge-based architecture separates representation from reasoning. A system capable of reflexive abstraction ("intelligent") reasons guided by the problem specification and by solving strategies. These are derived from higher levels of knowledge, representing principles of approach that are structured by even higher levels, containing hierarchical type theories. A possible interpretation of such hierarchies is: real time of the bottom levels - corresponding to primary knowledge/ behavior/ methods, is managed at upper levels - corresponding to concrete types/ strategies/ models, and abstracted on highest levels - corresponding to abstract types/ theories/ techniques.

An object-oriented simulation framework (Zeigler et al. 2000) permits the representation of different knowledge levels, each having a concept hierarchy, possibly abstraction/ symbolization/ structure leveled. 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. Knowledge levels are mathematical data/ operations types, approach/ solving strategies, simulation (description, representation, processing) and application concepts. The upper levels contain functions defined on the lower ones. Knowledge is represented as a hierarchical associative net of concepts: each level consists of clusters of related concepts.

Class-instance hierarchy can support symbolic operations; e.g., λ -calculus, extended to cope with program termination, i.e., partial functions, and evaluation. Symbolic formula is knowledge about expressions obtained by evaluation of some symbols (metaknowledge). For linear/ weakly nonlinear systems, algebra offers symbolic methods, to determine directly functional behavior; their instantiation result in the correspondent numerical methods.

To simulation, the hierarchical principle offers the advantage of adaptable 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. E.g., an unfinished design module is characterized by constraints regarding its interaction to other modules, resulting a model system, open to be interpreted, thus implemented, in different ways that derive from adequacy criteria.

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 and domain-specific calculus. Using constructive predicate logic, e.g., intuitionistic (Turner 1991), behavior or structure of the simulated system can be extracted from the proof.

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 learns by generating and validating models of the object-system. 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 ones; 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 or classes as well.

Knowledge hierarchy is not based on a form of 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 recursive functions defined for these objects. Interlevel relations can be interpreted as planning (top-down) and learning (bottom-up). A continuous model or a better approximation of continuity

by is needed to have the necessary conditions for conscience, what, together with adaptability, is necessary for intelligence.

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: induction - extensive knowledge at the lower level is transformed into intensive knowledge at the upper level, by non-reflexive abstraction (equivalence, isolation, emphasis, stationary approximation, idealization) and deduction - intralevel concept production (conditioning, association, stress, imitation).

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 - 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). A plan is a non-commutative system of declarative knowledge; extreme cases: commutative rule set, sequential procedure.

Human intelligence uses a reflexive, associative and prospective memory. Part of it is fast and volatile, performing interpreting operations; an other part is slow and persistent, storing the interpretation results (long-term memory). The volatile part is partly conscious (short-term memory), partly subliminal (work memory). Long-term knowledge is accessed through association to linguistic entities, transferred to the work memory, where interpretative operations, using their initial cognitive context, determine the coherence of the initial statement. A coherent interpretation causes surpassing the threshold of the consciousness; this calls for adding the statement to the short-term memory, then processing an automatic acquisition as well as a conscious treatment. Using this human architecture in simulation, we get closer to conscious simulation, thus extending adaptable to intelligent simulation, and to simulate intelligence. Sense is not just a composition of lexical entities any more, but corresponds to a contextual effect that can be modeled as a state transition of the system that recognizes the context-dependent language (linear-bounded-memory automaton) (Brecht et al. 1995). Based on knowledge accessibility of a subliminal level, a competitive (parallel) self-tuning models the semantic selection in the context. The state of the knowledge context acts as hypothesis set that gives priorities to the most coherent interpretations.

Hierarchical co-simulation

Intelligent systems call for hierarchical co-simulation of its hardware/ software parts, in the context of a unified representation of design and verification. Different hierarchy types structure the knowledge representation as well as the simulator itself: abstraction levels, block decomposition, concept hierarchies, symbolization degree, metaknowledge. In our opinion, a systematic hierarchical approach should follow next directions (Niculiu et al. 2000):

- object-oriented paradigm, which permits representing all hierarchy types, leading to compact, non-redundant, easy to modify models, for systems as well as processes, sequential or parallel;
- representation-inference dichotomy, that characterizes knowledge-based methods, enabling reflexive abstraction, thus, producing new knowledge starting with problem specification, following solving strategies, structured by approach principles and hierarchy types;
- unifying the simulation methods for the software/ hardware parts, combining complementary directions by object-orientation;
- formalizing representations in constructive mathematics style, advancing from correct specification to correct representation and, further, to correct simulation;
- multi-hierarchical simulation procedures expressed in an object-oriented knowledge-based framework, allowing recurrence for different hierarchy types.

Conclusions

We created a theoretical kernel for systems self-organizing by formal hierarchical descriptions. Unified representation for constructive and comprehensive activities, separated from the general methods of multi-hierarchical operation gives a new perspective on simulation. This permits further progress to simulated intelligence. We intend to develop an integrated programmable adaptable system for hardware/ software co-simulation. Further conclusions of this work, guiding our future research, are:

- Simulation is algorithmic theory.
- Simulated intelligence needs a hierarchy types theory.
- Knowledge hierarchies demand for an extended concept of algorithm, i.e., extending computability.

References

Brecht, W. 1995. *Theoretische Informatik*. Vieweg Verlag Braunschweig, Germany.

Kasch, F. and B. Pareigis1995. *Theoretische Informatik*. Fischer Verlag München, Germany.

Niculiu, T.; C. Aktouf; S. Cotofana,. 2000. "Hierarchical Interfaces for Hardware/ Software Systems". in Proceedings of the *European Simulation Multiconference* (Ghent, Belgium, May 23-26)..SCS Europe, Ghent Belgium , 647-655.

Turner, R. 1991. *Constructive Foundations for Functional Languages*. McGraw Hill, New York, NJ.

Zeigler B. et al., 2000. *Theory of Modeling and Simulation*, Academic Press, San Diego, CA.