On the Notion of Developmental Computing Machine

Antônio Carlos da Rocha Costa [1,2] **and Graçaliz Pereira Dimuro** [1]

[1]Programa de Pós-graduação em Informática – Escola de Informática
Universidade Católica de Pelotas, 96.010-000 Pelotas, RS, Brazil

[2]Programa de Pós-graduação em Computação – Instituto de Informática,
Universidade Federal do Rio Grande do Sul, 91.501-970 Porto Alegre, RS, Brazil.

`{rocha,liz}@atlas.ucpel.tche.br`

Abstract. *A developmental machine is a machine that is able to increase its structure and functionality as it operates in interaction with its environment. This paper initially explains the importance that the notion of machine development may have for Computer Science. Next, it attempts to identify the main reasons why the research community that is keen to the foundational issues lacks a serious concern with such issue. Then, it presents some fundamental problems that apparently have to be tackled by any approach to the notion of developmental computing machine. Finally, the paper introduces the initial elements of the domain-theoretic model of machine development that we are constructing.*

1. Introduction

SBC (Sociedade Brasileira de Computação), the Brazilian Computer Society, organized in May 2006 a workshop to establish the Grand Research Challenges of Computer Science in Brazil. The text that summarizes the results of the workshop is available from the society's website. The 4th of those grand challenges concerns the "Impacts on Computing of the Transition from Silicon to New Technologies", which rightly states that recent progress in the area of Molecular Biology may inspire new technological developments in Computer Science (CS).

This paper attempts to address the foundational issues raised by the 4th challenge, specially by the particular statement that "biological" technologies may impact CS. We focus mainly on the notion of development, which such "biological" technologies may bring to the domain of computing machines. We first point out that a clear notion of machine development is lacking in the epistemological framework of CS, and we identify in the limited conceptual horizon adopted by the classical Theory of Computation the reasons for such fault. Then, we summarize the current stage of development of our formal model of developmental computing machine. Finally, we remark that the impact of the introduction of a notion of developmental machine will not restrict itself just to the technical or methodological levels of computing: it will undermine the very conceptual foundations of CS.

2. On the Need of a Development Model for Computing Machines

This section explains why the notion of development is essential for computing machines, and why it is important that CS produces as soon as possible a formal model for developmental machines.

2.1. On the Notions of Development and Evolution of Complex Systems

Living beings, social organizations, and complex functional artifacts (e.g., complex computing systems) are similar in the difficulty of articulating adequate notions for their conceptual explications, and for formalizing them by mathematical means.

We take the viewpoint that two notions that can't be dismissed when considering such systems are the notions of development and evolution. *Development* is usually thought of as concerning modifications in individuals, leading them from states of lesser individual structural, functional and behavioral features to states of greater individual structural, functional and behavioral features, while *evolution* is thought of as concerning modifications in populations of individuals, similarly leading such populations from states of lesser collective structural, functional and behavioral features to states of greater collective structural, functional and behavioral features. Clearly, development and evolution should be seen as inter-related, so that the evolution of a population impacts the development of its individuals, and vice-versa.

We aim at applying Jean Piaget's explication of the notions of development and evolution of complex systems to the development and evolution of computing machines[1], i.e., we aim at connecting the concepts of machine development and machine evolution with concepts such as *autonomy*, *adaptation* and *operatory equilibration*. Such concepts refer to processes and structures that are still widely unexplored in Computer Science, so our work should be seen as placed in an epistemological space were a wide open theoretical scope still exists for any attempt at introducing formal models for such ideas.

We note that we are mainly concerned with the formal explication of the notion of machine development, the explication of machine evolution being left for future work. Also, we note that the machines with which we are concerned are computing machines, that is, machines whose main structural, functional and behavioral features refer to the processing of symbolic information, although we stress that we conceive them in tight interaction with environments characterized by both informational and material (i.e., physical) features. We also note that we do not aim, in the present paper, to present any foundational advances besides: (1) summarizing our view that the concepts elaborated by Piaget for the explication of the notion of (biological and psychological) development can support the establishment of a sensible notion of development for computing machines; and (2) showing that such notion of machine development can be given a suitable expression in the language of Domain Theory [Gierz et al. 2003].

Development is, in any of its possible conceptualization, a growing process, where the system increases (and reorganizes) its structure and functionality, thus requiring an increasing number of elements and capabilities. Computing models, such as the classical version of the Turing machine, have fixed structure and functionality, and lack adequate forms of interaction with the environment, and are thus unable to develop (see Sec. 2.2).

Computing systems that are able to go through developmental processes, which we call developmental machines [Costa and Dimuro 2005, Dimuro and Costa 2007], are systems required to exchange material items with the environment where they operate,

[1]See [Piaget 1971] for an account of Piaget's approach to general biological processes and structures, including an account of the notions of biological development and evolution, and [Piaget 1985] for an account of his approach to the psychological processes and structures involved in cognitive development.

besides exchanging informational items, so that they are able to increase the number and size of their components, thus becoming able to develop themselves, while working.

After the stages of the logical-mathematical computing models of information processing, like Turing's [Turing 1936], of psychological [Newell and Simon 1972] and sociological [Demazeau and Müller 1990] models of information processing, and after the population-based biological models of information processing [Eberhart et al. 2001], the case should be now very clear to CS that the notion of individual development has not yet been incorporated into the current repertoire of models of computing, and that providing for this incorporation can open an interesting array of possibilities [Costa 1993].

2.2. On the Limitations of the Classical Models of Computing Machines

This section briefly attempts to make clear that development is a notion that is not embedded in the classical models of information processing machines, which underlie the classical Theory of Computation, and that this fact impacted the intellectual tradition that derived from that theory, keeping it far away from the problem of machine development.

Take, for instance, the classical model of the Turing machine, in any of its forms, for example, the original form stated by Alan Turing [Turing 1936], or the form it acquired in the 1940's, as exposed by Kleene [Kleene 1952] or Rogers [Rogers 1967], which is essentially the one used until now, e.g., as in [Lewis and Papadimitriou 1997].

A Turing machine is a tuple $(Q, \Sigma, \Gamma, q_0, F, D, \Delta)$ where Q is a finite set of states, Σ is a finite set of input symbols, Γ is a finite set of internal memory (tape) symbols, q_0 is the initial state in which the machine starts its operation, F is the set of final states, D is the set of possible directions of movements of the access point of the internal memory of the machine, Δ is the configuration transition relation of the machine. The most important features of such structure, which forbid the support of any kind developmental process, is the fixed nature of its constituent sets (Q, Σ, Γ) and transition relation (Δ).

Moreover, a complementary feature reinforces the structural and functional fixity of the model, namely, that Turing machines operate as closed systems, while computing. This is so because Turing machines, as any other classical model of computing machines, operate by computing mathematical functions and, thus, cannot support interaction. And in turn, this is so, due to the simple fact that a mathematical function cannot be applied to an argument that is not well defined (and cannot even be defined if the elements of its domain are not known).[2] Lack of interaction means lack of possibility of exchange of informational and material items with the environment and, thus, lack of possibility of development while functioning.

2.3. An Aside: The Essential Advancement Provided by John von Neumann (or, Why the Theory of Computation is not the Theory of the Real Computers)

Part of the condition mentioned above, that the classical model of computing machine is a non-interactive one, was overcome by John von Neumann's proposal of a computer

[2]A usual misunderstanding often arises, as a counter-argument, at this point of the exposition, namely, that any sequence of inputs could be completely encoded in the Turing machine tape and thus the computation could happen by simulating the interaction with the user/environment by using that pre-encoded sequence of inputs. But this presumed counter-argument fails essentially because it misses the point that a pre-encoded sequence of inputs requires that the sequence of inputs be defined before the interaction starts, precisely the requirement that any true interaction does not impose on its participants!

architecture that besides incorporating all the important features of the Turing machine model, introduced the possibility of interaction, by allowing input and output operations to occur interspersed within the computation.

Such possibility of interaction during computation, that is completely lacking in any of the classical models, is the true foundation of the usability of real computers, because it allows the computing machine to go beyond the mere computation of mathematical functions. In particular, it allows the computation to start and produce initial results even before any input data or command is defined (or even known) by the person or environment with which the computing machine is interacting.

In other words, the classical version of the Theory of Computation, developed by Turing, Church, Kleene, Post and others, is not the theory of the real computers that we deal with since von Neumann. It is, just, the theory of one particular way in which we are able to use real computers, namely, for computing mathematical functions.

For the most part, programming real computers means programming a coordination task, where the actions of the computer and of its users/environment are to be orchestrated in order that a service be performed, either by the computer to the environment, or vice-versa. This is far beyond the scope of the intuitive (and even theoretical) goals of the computing task, as originally understood by those that founded the Theory of Computation. It clearly goes far beyond any semantics they assigned to the word computation.[3]

2.4. On the Limitations of the Contemporaneous Models of Computing Machines

All of the contemporaneous models of computing machines also lack adequate support for the notion of development. For instance, models of reactive systems expressed as calculus of processes (e.g., [Milner 1989, Hoare 1985]), go far ahead of the classical models by incorporating the notion of interaction, but still fall short of any developmental model, by being organized around a fixed transition system, or a fixed set of algebraic axioms.

Development implies the creation of innovation in the rules of behavior, and fixed sets of rules are not able to represent it adequately. Also, the interactive versions of Turing machines, like the Persistent Turing Machine [Goldin et al. 2006], and the lazy notions of computation [Escardo 1993], have precisely that same feature of a fixed structure, determined by fixed sets of input and output data, control mechanisms or axiomatic basis.

In fact, the mighty slogan created by Peter Wegner, that ”interaction is more powerful than algorithms” [Wegner 1997] goes just half the way towards the end of the story: only interaction understood as exchange of both informational and material items with the environment is the full realization of the notion of interaction as it is required by CS, and witnessed by domains like Biology, Psychology and Sociology (including Economics).

It seems that CS is ready to take this step towards a developmental model of computing machines, diminishing the gap between computational models and the other models of complex systems, but to do so it must put aside the tenets that the classical version

[3]With the obviously possible exceptions of Alan Turing and his concerns about machine intelligence, which were never explicitly incorporated into his computation-theoretic studies [Copeland 2004], and of John von Neumann and his concerns with complex automata, which however seems to have not overcome the limitations of his purely informational conception of the internal functioning of automata [Neumann 1966].

of the Theory of Computation imposed on it, including the tenet that Peter Wegner mentioned, that every computation is algorithmic.

2.5. On Importance of the Piagetian Notion of Development

Biology has constructed several models of development along its history. Choosing one to be the reference for a model of development in CS may take a long time. Such a model should not be "too biological", in the sense of depending excessively on the biomolecular properties of the basic components of the living beings, because it may well happen that the adequate technology for the developmental machines be not "celular" in nature (although this is one of the best bets one can make, at the moment). The model should also not be too general, as e.g. the early "cybernetic" models of biological organisms [Wiener 1948], so as to abstract away elements that are essential to any developmental system. Nor should it be too oriented towards "information processing", like the classical models that served as foundation for the good old fashioned Artificial Intelligence [Newell 1980, Newell 1982], so as to simply ignore the possibility and relevance of material exchanges of the systems with their environments.

We have chosen as the conceptual basis of our work a development model that seems to be general enough, so as to be free from commitments to biomolecular mechanism, yet is detailed enough in order to capture some general structures and processes that seem to be mandatory in any model of development of complex systems. We have chosen the general development model that Jean Piaget abstracted from his studies of both biological [Piaget 1971] and psychological systems [Piaget 1985], and that guided his reflections about the notion of evolution in Biology and Psychology [Piaget 1979, Piaget 1980].

We refrain from summarizing here the Piagetian model of development, for the sake of space, and content ourselves to address it as we present the results that we have achieved in the construction of our model of developmental computing machines.

2.6. Another Aside: On the Role of the Objectivistic Notion of Machine Intelligence

Since its inception, Artificial Intelligence has oscillated between aiming at the simulation of human mind, and a more bold aim of constructing really intelligent artifacts [Searle 1980], often oscillating around the simpler idea of using psychological or sociological metaphors to support the programming of computer systems. In all cases, however, the stance adopted by the AI researcher is that of the creator of the (real, simulated or metaphorical) mind of the artifact.

From the constructivist point of view [Piaget 2001], intelligence is not something that can simply be added to a complex system, from outside. Intelligence, wherever possible, is the result of an internal need of the system, the need for equilibrium (and stability) in the system's functional interaction with the environment.

A functional interaction involves more than a material interaction: the latter is exchange of material substances and components with the environment. Functional interaction is exchange of actions and information. An internal need is the need of a structure or process, that arises in a system through the operation of an internal mechanism that is activated if and only the system is put into situations where the lack of that process or structure menaces to jeopardize the continuity of the system's internal functioning, and possibly also its adaptation the environment.

Intelligence is a structure that results from an adaptation process to the environment, that is, a structure that is produced by a combination of an evolution process (at the population level) and a development process (at the individual level). Its purpose is the regulation of the system's functional interaction with the environment, aiming at keeping it equilibrated and stable [Piaget 2001]. Any complex system that is able to regulate its functional interaction with its environment, keeping itself equilibrated and stable, is necessarily endowed with a structure responsible for such regulation, called its intelligence.

Computing systems are systems that are able to regulate their functional interaction with the environment, so computing systems are naturally (that is, by virtue of their constitution and organization) endowed with an intelligence, that is, its very project contains that structure in it, otherwise the regulation of the functional interactions would not be possible. We call objectivistic, or naturalistic, such point of view about the notion of machine intelligence [Costa 1993].

It should be clear, then, that from this objectivistic viewpoint, the notion of machine intelligence is a notion that has to play a central role in any Theory of Computation that adequately copes with real computing machines. Otherwise, the structures and processes that regulate the functional interaction of the computing machine with its environment will be improperly abstracted away from the theoretical account of that machine.

Thus, the main purpose of Artificial Intelligence should not be cast in terms of "creating" machine intelligences, simply because machine intelligences are already there in every computing system, but should be cast in terms of establishing the conditions for the evolution and development of such machine intelligences.

Moreover, given the current technological constraints that do not contemplate the possibility of material exchanges becoming part of the functional interaction of computing machines with their environments, it seems that Artificial Intelligence should also engage with its specific conceptual frameworks and methodologies in this task of furthering the underlying technology of computing systems, making it go beyond our current silicon based one, in order to open the possibility of integrating material exchanges into the functional interactions that are based purely on informational exchanges.

The possibility of having functional interactions based on material and informational exchanges will allow computing machines to enormously improve the quality, strength and scope of their functional exchanges, compared with what they are able to do presently, because it will allow the furthering of the regulation structures and processes that constitute their machine intelligence [Costa 1993].

3. Sketch of a Development Model for Computing Machines

In this section, we briefly sketch the main elements of the model of developmental computing machines that we are constructing. Most of the intuitions behind our formal notion of developmental computing machine can be found in [Costa 1993].

The mathematical structure that we use to build our development model is based on the so-called *domain* [Gierz et al. 2003], which was first introduced by Scott [Scott 1970] to give denotational semantics to programming languages. Domains are partially ordered sets $(D, \sqsubseteq)$ with some additional properties, and where the order $\sqsubseteq$ relates two objects according to the degree of achievement of their *construction*. That is, given two objects $x, y \in D$, we say that $x \sqsubseteq y$ if and only if the set of features that characterize y includes all those that characterize x and possibly some additional ones, so that y is more *complete* (more *developed*, as we say in this context) than x.

The domains that we have chosen as the basis of our model are the so-called *coherence spaces* [Girard 1987]. A coherence space is a collection of sets ordered under the inclusion relation, satisfying some additional requirements. Each object of the domain, called *coherent set*, is a set of tokens that satisfies a certain coherence condition. An important property of coherence spaces is that the objects that are considered *finite* (compact) from the point of view of Domain Theory are also finite from the point of view of Set Theory, which is not always true for other kinds of domains. This is specially important when modeling domains of finite machines, as in the case of this paper.

Every domain has a *least element* $\perp$, which is the least developed of all objects, and a set of *total* elements, which are completely developed. A *chain* of partial objects $x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ may be understood as a construction (development) process, where each object is an intermediate stage of the process. Any such chain in a domain has a limit (least upper bound), which represents the final result of such development process. When the limit is a total object that means that no further improvement on it can be obtained by that development process.

We note, however, that in Piaget's conception, development is a process performed by two kinds of operations, the so-called equilibration operations [Piaget 1985]. The first kind of equilibration operation, called *minor equilibration*, contributes to development by increasing the quality of the systems' behavioral, organizational and functional features without altering the so-called *developmental level* of the system: it expands the scope of applications of each of the operations present in the system's structure, in an incremental way, without altering the operatory properties of that structure as a whole. The second kind of equilibration operation, called *major equilibration*, contributes to development by modifying substantially the system's structure, expanding the set of operations and relations responsible for the systems' features, leading the system to a new developmental level.

So, we have to formalize the development model in the following way. We consider domains of possible construction stages of a machine, where each object represents the machine in a particular possible stage of its development.

Minor equilibrations are modeled as *chains* of stages of machine development within *domains of stages of machine development*. Each such domain represents a *development level* of the overall development of the machine. Formally, we operate with domains that are isomorphic to coherence spaces, so that stages of machine development can be seen as coherent sets, and machine developments produced by minor equilibrations as chains of coherent sets.

Major equilibrations are modeled as *chains of domains* within the *ordered category of domains of stages of machine development*, where the order relation between development levels models the relative expansion of a machine's structure. An essential feature of a major equilibration is that every object present in the previous level of development is preserved in the next level of development together with new elements that may be added on the basis of new combinations of previous elements. Thus, major

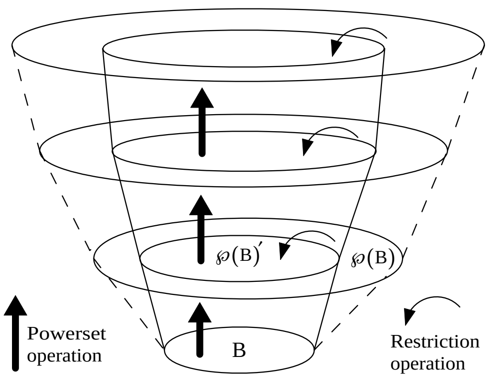

Figure 1. An intuitive view of our formal account of the operation of major equilibration.

equilibrations involve a “powerset-like” operation that needs to be represented in the formal model. So, coherence spaces [Girard 1987] are also adequate for the purpose of our work, because their powerset operation, necessary for modeling the major equilibration operation, is a functor in the category of coherence spaces and continuous functions.

In fact, major equilibrations are modeled here, and in [Dimuro and Costa 2007], in a tentative way, by the construction process introduced in [Dimuro 1998, Dimuro et al. 2000], where: (1) a powerset operation is applied to an initial coherence space $\mathbf{B}$, obtaining a new coherence space $\wp(\mathbf{B})$, ensuring that $\mathbf{B}$ is embedded in it; (2) restriction operations are performed, to guarantee that a subsystem $\wp(\mathbf{B})'$ is well behaved with respect to certain criteria.

Figure 1 illustrates intuitively the first few major equilibrations of the domain $\mathbf{B}$. The chain of restricted domains obtained as the result of a major equilibration process is the *domain-theoretic support* for the whole development process of the computing machine, which is being modeled by that structure. Each of the domains in that support is a *level of development*, achieved by a major equilibration operation, and within which minor equilibration operations occur.

4. On the Formal Presentation of the Basic Development Model

4.1. Coherence Spaces and the Minor Equilibration Process

We take the simplifying view that a computing machine, in any stage of any level of its development, can be seen as a dynamical system $m = (S, \delta)$, where S is a set of states and $\delta \subseteq S \times S$ is a (partial) transition relation defined on S. We assume, then, that a computing machine has a non-deterministic dynamics, in general. In the particular cases in which δ is a (partial) function, the dynamics is deterministic. The set of all transition relations is denoted by Δ_S, while $\widetilde{\Delta}_S \subseteq \Delta_S$ denotes the set of all transition functions.

The development of a computing machine, given by a succession of minor equilibration operations, is a chain in the set $\mathbb{M} = \{S\} \times \Delta_S$ of stages of machine development, partially ordered by a development relation $\sqsubseteq \subseteq \mathbb{M} \times \mathbb{M}$, defined as:

$$\forall m_1 = (S, \delta_1), m_2 = (S, \delta_2) \in \mathbb{M} : m_1 \sqsubseteq m_2 \Leftrightarrow \delta_1 \subseteq \delta_2. \quad (1)$$

The structure $\mathbf{M} = (\mathbb{M}, \sqsubseteq, m_0)$ is a domain in the general sense of Scott's theory [Gierz et al. 2003], as explained in Sect. 3, where $m_0 = (S, \delta_0)$ is the least element of the domain, representing the initial stage in the development process ($\delta_0 = \emptyset$).

Analogously, we obtain the domain $\widetilde{\mathbf{M}} = (\widetilde{\mathbb{M}}, \sqsubseteq, \widetilde{m}_0)$ of stages of development of deterministic computing machines, where $\widetilde{\mathbb{M}} = \{S\} \times \widetilde{\Delta}_S$ and $\widetilde{m}_0 = (S, \widetilde{\delta}_0)$, with $\widetilde{\delta}_0 = \emptyset$.

As discussed in Sect. 3, a coherence space is a special kind of domain:

Definition 4.1 *A* coherence space *is a structure* $\mathbf{A} = (\mathbb{A}, \subseteq)$*, where* $\mathbb{A}$ *is a of family of (coherent) sets ordered under the inclusion relation, with the following properties: (i)* down-closure*: if* $a \in \mathbf{A}$ *and* $a' \subseteq a$*, then* $a' \in \mathbf{A}$*; (ii)* binary completeness*: if* $X \subseteq \mathbf{A}$ *and if* $\forall a, a' \in X : (a \cup a') \in \mathbf{A}$*, then* $\bigcup X \in \mathbf{A}$*.*

From any coherence space $\mathbf{A}$, it is possible to find its *web* [Girard 1987], the very basic structure from which $\mathbf{A}$ originated, which is given by: $\mid \mathbf{A} \mid = (\{\alpha \mid \{\alpha\} \in \mathbf{A}\}, \approx)$, where any α is called a *token* and $\approx$ is a reflexive and symmetric relation, called *coherence relation*, defined between tokens by: $\alpha \approx \alpha' \Leftrightarrow \{\alpha, \alpha'\} \in \mathbf{A}$. A coherent set $a \in \mathbf{A}$ is then a set of pairwise coherent tokens.

In the following, we introduce representations of domains of stages of development of (deterministic and non-deterministic) machines in terms of coherence spaces. For that, Prop. 4.2 and Prop. 4.3 establish the isomorphisms between those two kinds of domains, as proved in [Dimuro and Costa 2007].

Proposition 4.2 *Let* $\widetilde{\mathbb{M}} = \{S\} \times \widetilde{\Delta}_S$ *be the set of stages of development of a* deterministic *machine, where* $\widetilde{\Delta}_S$ *is the set of* (partial) transition functions $\widetilde{\delta}$ *on* S*. Then: (i)* $\widetilde{\mathbf{D}} = (\widetilde{\Delta}_S, \subseteq)$ *is a coherence space, with web* $\mid \widetilde{\mathbf{D}} \mid = (S \times S, \approx)$*, and coherence relation* $\approx \subseteq (S \times S) \times (S \times S)$ *given by:* $(x, y) \approx (x', y') \Leftrightarrow ((x, y) = (x', y') \vee x \neq x')$*; (ii) The structure* $\widetilde{\mathbf{M}} = (\widetilde{\mathbb{M}}, \sqsubseteq, \widetilde{m}_0)$*, where* $\sqsubseteq$ *is defined analogously to* (1) *and* $\widetilde{m}_0 = (S, \widetilde{\delta}_0 = \emptyset)$*, is isomorphic to* $\widetilde{\mathbf{D}}$*.*

Proposition 4.3 *Let* $\mathbb{M} = \{S\} \times \Delta_S$ *be the set of stages of development of a* non-deterministic *machine, where* Δ_S *be the set of* (partial) transition relations δ *on* S*. Then: (i)* $\mathbf{D} = (\Delta_S, \subseteq)$ *is a coherence space, with the trivial coherence relation* $\approx \subseteq (S \times S) \times (S \times S)$*, such that* $(x, y) \approx (x', y')$*, for all* $(x, y), (x', y') \in S \times S$*. (ii)* $\mathbf{M} = (\mathbb{M}, \sqsubseteq, m_0)$*, with* $\mathbb{M} = \{S\} \times \Delta_S$*,* $m_0 = (S, \delta_0 = \emptyset)$ *and* $\sqsubseteq$ *as given in* (1)*, is isomorphic to* $\mathbf{D}$*.*

In the following, we shall abuse the notation and refer to both domains of stages of machine development $\mathbf{M}$ and $\widetilde{\mathbf{M}}$ as coherence spaces.

However, as detailed in [Piaget 1985], the operations of minor and major equilibrations should be analyzed in terms of a set of internal and external *development factors* (e.g., some condition of behavior enabling, some functional requirement, etc.) that determine the way they work.

So, in our model, we assume that the universe of $\mathbf{M}$ is restricted by a set I of internal development factors. Determining the family of developmental stages that are admissible for I, we could define a developmental structure $(\mathbb{M}_I, \sqsubseteq_I, m_0)$, where $\mathbb{M}_I$ is the subset of stages of machine development that are compatible with I, and $\sqsubseteq_I \subseteq \mathbb{M}_I \times \mathbb{M}_I$ preserves the compatibility character during the development process. I can be seen as a set of criteria for the *internal* consistency of the admissible structures of $\mathbf{M}$.

Similarly, the influence of the external environment must be considered through a set of *external* development factors E. I and E lead the development process in $\mathbf{M}$ towards a final, total developmental stage, which is the limit of the development chain in that domain.

A formal structure able to model in a better way the *domain of development stages* of a computing machine (at a given development level) would be, thus, a structure $(\mathbb{M}_{I,E}, \sqsubseteq_{I,E}, m_0)$, where $\mathbb{M}_{I,E}$ is the subset of development stages of $\mathbf{M}$ that are compatible with both I and E, and the development relation $\sqsubseteq_{I,E}$ respects both types of development factors.

At this point, one sees that the domain of development stages presents a double (internal and external) structure. Then, formally, it is necessary to deal with the *bi-structured coherence spaces* [Dimuro 1998] in order to represent those domains:

Definition 4.4 *A bi-structured coherence space is a system* $\mathbf{A} = (\mathbb{A}; \Sigma^I_{\mathbb{A}}; \Sigma^E_{\mathbb{A}})$*, with signature* $\langle \mu_I; \mu_E \rangle$*, where: (i)* $\mathbb{A} \neq \emptyset$ *is the universe of a coherence space; (ii)* $\Sigma^I_{\mathbb{A}} = (\subseteq_{\mathbb{A}}, \{g_{\mathbb{A}_l} : \mathbb{A}^{\mu_I(l)} \to \mathbb{A}\}_{l \in L})$ *is the* internal structure *of* $\mathbf{A}$*, determined by the construction order (the inclusion relation* $\subseteq_{\mathbb{A}}$ *defined on* $\mathbb{A}$*), together with functions* $g_{\mathbb{A}_l}$*, which represent the* internal factors *of the construction process, with arities given by* $\mu_I : L \to \mathbb{N}$*; (iii)* $\Sigma^E_{\mathbb{A}} = (\{f_{\mathbb{A}_k} : \mathbb{A}^{\mu_E(k)} \to \mathbb{A}\}_{k \in K})$ *is the* external structure *of* $\mathbf{A}$*, determined by the functions* $f_{\mathbb{A}_k}$*, which represent the* external factors *of the construction process, with* $\mu_E : K \to \mathbb{N}$.

The category $\mathbf{BSCS}$ has bi-structured coherence spaces as objects and strong homomorphisms as morphisms [Dimuro 1998].

Given that the development factors may be defined as functions on $\mathbb{M} = \{S\} \times \Delta_S$, it is possible to take a domain $\mathbf{M} = (\mathbb{M}, \sqsubseteq, m_0)$ of stages of development and make explicit those factors in its structure.

Let $\mathbf{M}_{I,E} = (\mathbb{M}; \Sigma^I_{\mathbb{M}}; \Sigma^E_{\mathbb{M}})$ be the extended domain, where $\Sigma^I_{\mathbb{M}}$ is the internal structure (the development order $\subseteq$ and the internal development factors I), and $\Sigma^E_{\mathbb{M}}$ is the external structure (the external development factors E). Then, considering the isomorphism between the domain $\mathbf{M}$ of stages of machine development and the coherence space $\mathbf{D} = (\Delta_S; \subseteq)$, stated in Prop. 4.2, it holds that the bi-structured domain $\mathbf{M}_{I,E} = (\mathbb{M}; \Sigma^I_{\mathbb{M}}; \Sigma^E_{\mathbb{M}})$ is isomorphic to the bi-structured coherence space $\mathbf{D} = (\Delta_S; \Sigma^I_{\Delta_S}; \Sigma^E_{\Delta_S})$.

For our purpose, however, we need to get the bi-structured system $\mathbf{M}^*_{I,E} = (\mathbb{M}_{I,E}; \Sigma^I_{\mathbb{M}_{I,E}}; \Sigma^E_{\mathbb{M}_{I,E}})$, where the subset $\mathbb{M}_{I,E} \subseteq \mathbb{M}$ has only developmental stages compatible with both I and E.

For that, we use the representation of bi-structured domains as bi-structured coherence spaces, and define in the following the notion of a sub-system of a bi-structured coherence space which is well-behaved with respect to the internal and external factors.

Let $\mathbf{A} = (\mathbb{A}; \Sigma^I_{\mathbb{A}}; \Sigma^E_{\mathbb{A}})$ and $\mathbf{B} = (\mathbb{B}; \Sigma^I_{\mathbb{B}}; \Sigma^E_{\mathbb{B}})$ be bi-structured coherence spaces, both of the same signature $\langle \mu_I; \mu_E \rangle$, as introduced in Def. 4.4.

Definition 4.5 $\mathbf{A}$ *is said to be a sub-system (restricted by a function* $p : \mathbb{B} \to \mathbb{B}$*) of* $\mathbf{B}$*, denoted by* $\mathbf{A} = p(\mathbf{B})$*, if and only if: (i)* $\mathbb{A} = p[\mathbb{B}] = \{p(b) \mid b \in \mathbb{B}\}$*; (ii)* $\forall l \in L : g_{\mathbb{A}_l} = g_{\mathbb{B}_l} |_{\mathbb{A}^{\mu_I(l)}}$*, i.e.,* $g_{\mathbb{A}_l}$ *is the restriction of* $g_{\mathbb{B}_l}$ *to* $\mathbb{A}$*; (iii)* $\forall k \in K : f_{\mathbb{A}_k} = f_{\mathbb{B}_k} |_{\mathbb{A}^{\mu_E(k)}}$*, i.e.,* $f_{\mathbb{A}_k}$ *is the restriction of* $g_{\mathbb{B}_k}$ *to* $\mathbb{A}$.

We observe that the sub-system $\mathbf{A}$ mentioned in Def. 4.5 is not necessarily a bi-structured coherence space.

The operation of *closure*, with respect to a given sub-system $p(\mathbf{A})$, of a function $h_{\mathbb{A}_i} = \mathbb{A}^{\mu(i)} \to \mathbb{A}$ that belongs to either the internal or the external structure of the bi-structured coherence space $\mathbf{A}$ completes the graph of the function, ensuring that the function is well-behaved in that sub-system (the image of every coherent set is a coherent set) [Dimuro et al. 2000].

The closure of $h_{\mathbb{A}_i} = \mathbb{A}^{\mu(i)} \to \mathbb{A}$, defined on a bi-structured coherence space $\mathbf{A}$, with respect to a sub-system $p(\mathbf{A})$, itself closed for the intersection operation, is the function $\hat{h}^{p[\mathbb{A}]}_{\mathbb{A}_i} : \mathbb{A}^{\mu(i)} \to \mathbb{A}$, defined by:

$$\hat{h}^{p[\mathbb{A}]}_{\mathbb{A}_i}(X_1, \ldots, X_{\mu(i)}) = \begin{cases} \bigcap\{Y \in p[\mathbb{A}] \mid h_{\mathbb{A}_i}(X_1, \ldots, X_{\mu(i)}) \subseteq Y\} & \text{if } X_1, \ldots, X_{\mu(i)} \in p[\mathbb{A}]^{\mu(i)}, \\ h_{\mathbb{A}_i}(X_1, \ldots, X_{\mu(i)}) & \text{otherwise.} \end{cases}$$

Given a domain $\mathbf{B}$ and an operation p on its objects, we call *regulation* of $\mathbf{B}$, with respect to p, the domain operation R_p that closes every internal and external function of $\mathbf{B}$ with respect to the sub-system $p[\mathbf{B}]$.

Definition 4.6 *The bi-structured coherence space* $\mathbf{A}$ *is* obtained by regulation *from the bi-structured coherence space* $\mathbf{B}$, *with respect to an operation* $p : \mathbb{B} \to \mathbb{B}$, *denoted by* $\mathbf{A} = R_p(\mathbf{B})$, *if and only if: (i)* $\mathbb{A} = \mathbb{B}$*; (ii)* $\forall l \in L : g_{\mathbb{A}_l} = \hat{g}^{p[\mathbb{B}]}_{\mathbb{B}_l}$*; (iii)* $\forall k \in K : f_{\mathbb{A}_k} = \hat{f}^{p[\mathbb{B}]}_{\mathbb{B}_k}$*; where* $\hat{g}^{p[\mathbb{B}]}_{\mathbb{B}_l}, \hat{f}^{p[\mathbb{B}]}_{\mathbb{B}_k}$ *are the closures of* $g_{\mathbb{B}_l}$ *and* $f_{\mathbb{B}_k}$ *with respect to* $p[\mathbf{B}]$.

Let $\mathbf{M}_{I,E} = (\mathbb{M}; \Sigma^I_{\mathbb{M}}; \Sigma^E_{\mathbb{M}})$ be the bi-structured coherence space that represents a given level of machine development, and $\mathbf{M}^*_{I,E} = (\mathbb{M}_{I,E}; \Sigma^I_{\mathbb{M}_{I,E}}; \Sigma^E_{\mathbb{M}_{I,E}})$ be the domain of the development stages that are admissible for the development factors I and E.

Finally, to make our model fully compatible with Piaget's notion of development, we require that the development factors be such that $\mathbf{M}^*_{I,E}$ is a sub-system of $\mathbf{M}_{I,E}$, restricted by the operation $* : \mathbb{M} \to \mathbb{M}$, such that $*[\mathbb{M}] = \mathbb{M}_{I,E}$ (that is, $*$ restricts the universe $\mathbb{M}$ to its part $\mathbb{M}_{I,E}$ that is admissible to both I and E). Also, all internal and external functions of $\mathbf{M}_{I,E}$ must be well-behaved in $\mathbb{M}_{I,E}$, that is, $\mathbf{M}_{I,E}$ be obtained by regulation with respect to the operation $*$.

Formally, we state the following *requirements* for the development factors:

(i) $\mathbf{M}^*_{I,E}$ is a subsystem of $\mathbf{M}_{I,E}$ that is itself a coherence space; and
(ii) $\mathbf{M}_{I,E} = R_*(\mathbf{M}_{I,E})$, that is, $\mathbf{M}_{I,E}$ is a fixpoint of R_*.

4.2. Coherence Space Constructors and the Major Equilibration Process

As explained in Sect. 3, a total stage of machine development, in a given level, represents a fully developed machine, with respect to the level of development (bi-structured coherence space) that is being considered. Only a *major equilibration operation* can embed the developmental machine into a higher-level bi-structured coherence space (the domain corresponding to a higher level of development) that allow further developments.

Operations of major equilibration can be modeled as coherence space transformations performed simultaneously on the universe and on the internal and external structures (called *global domain transformations* in [Dimuro 1998]). Those transformations are functors in the category $\mathbf{BSCS}$.

The constructors of Coherence Spaces [Girard 1987] may be extended to bi-structured coherence spaces. Some of then may have interesting interpretations when acting on a developmental machine domain.

For example, let $\mathbf{D} = (\Delta_S; \Sigma^I_{\Delta_S}; \Sigma^E_{\Delta_S})$ and $\mathbf{D}' = (\Delta_{S'}; \Sigma^I_{\Delta_{S'}}; \Sigma^E_{\Delta_{S'}})$ be the bi-structured coherence spaces isomorphic to developmental machines $\mathbf{M}_{I,E}$ and $\mathbf{M}'_{I,E}$. Considering the *tensor* operator $\otimes$ applied to $\mathbf{D}$ and $\mathbf{D}'$, one obtains a bi-structured coherence space $\mathbf{D} \otimes \mathbf{D}'$ determined by the web $((S \times S) \times (S' \times S'), \approx_\otimes)$, with

$$((x_1, y_1), (x'_1, y'_1)) \approx_\otimes ((x_2, y_2), (x'_2, y'_2)) \Leftrightarrow ((x_1, y_1) \approx_S (x_2, y_2) \ \wedge \ (x'_1, y'_1) \approx_{S'} (x'_2, y'_2)),$$

where $\approx_S$ and $\approx_{S'}$ are the coherence relations defined on the original webs $(S \times S, \approx_S)$ and $(S' \times S', \approx_{S'})$, respectively (see Prop. 4.2).

The domain $\mathbf{M}_{I,E} \otimes \mathbf{M}'_{I,E}$ constitutes a kind of "grid" of two developmental machines (i.e., two machines operating in parallel, with local transition functions).

In the following, let $\mathbf{A} = (\mathbb{A}; \Sigma^I_{\mathbb{A}}; \Sigma^E_{\mathbb{A}}), \mathbf{B} = (\mathbb{B}; \Sigma^I_{\mathbb{B}}; \Sigma^E_{\mathbb{B}}) \in \mathbf{BSCS}$ have both the same signature $\langle \mu_I; \mu_E \rangle$, as introduced in Def. 4.4. The *powerset constructor* of bi-structured coherence spaces is the main constructor of our model, since it supports the major equilibration operation, as explained in Sect. 3. This operator generates a new coherence space, whose universe is the powerset of the original one, so that the original universe is embedded in the new universe, and also extends the inclusion order, as well as the internal and the external functions to operate over sets of coherent sets.

Definition 4.7 *A powerset constructor is a map* $t \equiv (t_u; t_{\Sigma_I}; t_{\Sigma_E}) : \mathbf{BSCS} \to \mathbf{BSCS}$, *defined by* $t\mathbf{A} = (t_u\mathbb{A}; t_{\Sigma_I}\Sigma^I_{\mathbb{A}}; t_{\Sigma_E}\Sigma^E_{\mathbb{A}})$, *where:*

(i) t_u *is a powerset universe constructor such that* $t_u\mathbb{A} = \wp(\mathbb{A})$*;*

(ii) $t_{\Sigma^I} = \langle t_\subseteq, t^I_F \rangle$ *is the internal structure constructor, determined by the order constructor* $t_\subseteq$ *and the internal function constructor* t^I_F*, such that* $\langle t_\subseteq, t^I_F \rangle \Sigma^I_{\mathbb{A}} = (t_\subseteq \subseteq_{\mathbb{A}}, t^I_F\{g_{\mathbb{A}_l} : \mathbb{A}^{\mu_I(l)} \to \mathbb{A}\}_{l \in L})$*, where:*

(a) $t_\subseteq \subseteq_{\mathbb{A}} = \subseteq_{\wp(\mathbb{A})}$*, that is,* $t_\subseteq$ *transforms the inclusion relation defined on* $\mathbb{A}$ *into the one defined on* $t_u\mathbb{A} = \wp(\mathbb{A})$*, and*

(b) $\forall l \in L, g_{\mathbb{A}_l} \in [\mathbb{A}^{\mu_{in(l)}} \to \mathbb{A}] : g_{\mathbb{A}_l} \mapsto (t^I_F\, g)_{(t_u\mathbb{A})_l} \in [t_u\mathbb{A}^{\mu_{in(l)}} \to t_u\mathbb{A}]$*, where* $(t^I_F\, g)_{(t_u\mathbb{A})_l}$ *is the natural extension of* $g_{\mathbb{A}_l}$ *to* $t_u\mathbb{A}$*:* $(t^I_F\, g)_{(t_u\mathbb{A})_l}(X_1, \ldots, X_{\mu_I(l)}) = \{g_{\mathbb{A}_l}(x_1, \ldots, x_{\mu_I(l)}) \in \mathbb{A} \mid x_1 \in X_1, \ldots, x_{\mu_I(l)} \in X_{\mu_I(l)}\}$*, for each* $x_1, \ldots, x_{\mu_I(l)} \in \mathbb{A}$ *and* $X_1, \ldots, X_{\mu_I(l)} \in t_u\mathbb{A}$*;*

(iii) $t_{\Sigma^E} = \langle t^E_F \rangle$ *is an external structure constructor, determined by the external function constructor* t^E_F*, such that* $\langle t^E_F \rangle \Sigma^E_{\mathbb{A}} = t^E_F\{f_{\mathbb{A}_k} : \mathbb{A}^{\mu_E(k)} \to \mathbb{A}\}_{k \in K}$*, where* $\forall k \in K, f_{\mathbb{X}_k} \in [\mathbb{X}^{\mu_E(k)} \to \mathbb{X}] : f_{\mathbb{X}_k} \mapsto (t^E_F f)_{(t_u\mathbb{X})_k} \in [t_u\mathbb{X}^{\mu_{ex(k)}} \to t_u\mathbb{X}]$*, with* $(t^E_F f)_{(t_u\mathbb{X})_k}$ *being the natural extension of the function* $f_{\mathbb{X}_k}$ *to* $t_u\mathbb{X}$*, whose definition is analogous to* **(ii)(b)**.

Let $\mathbf{D} = (\Delta_S; \Sigma^I_{\Delta_S}; \Sigma^E_{\Delta_S})$ be the bi-structured coherence space that is isomorphic to a domain $\mathbf{M}_{I,E} = (\mathbb{M}; \Sigma^I_{\mathbb{M}}; \Sigma^E_{\mathbb{M}})$ of a developmental machine. Applying the powerset constructor to $\mathbf{D}$, the resulting bi-structured coherence space $\wp(\mathbf{D}) = (\wp(\Delta_S); \Sigma^{I'}_{\wp(\Delta_S)}; \Sigma^{E'}_{\wp(\Delta_S)})$, where $\Sigma^{I'}_{\wp(\Delta_S)}$ and $\Sigma^{E'}_{\wp(\Delta_S)}$ are the internal and external functions generated in the construction, has a web given by $\mid \wp(\mathbf{D}) \mid = (\Delta_S, \approx)$, where $\approx$ is the trivial coherence relation, that is, $\delta \approx \delta'$, for all $\delta, \delta' \in \Delta_S$. Observe that $\wp(\mathbf{D})$ represents (is isomorphic to) the powerset-like domain $\wp(\mathbf{M}_{I',E'}) = (\{S\} \times \wp(\Delta_S); \Sigma^{I'}_{\mathbb{M}}; \Sigma^{E'}_{\mathbb{M}})$.

The main feature of $\wp(\mathbf{M}_{I',E'})$ is that its objects are dynamical systems composed by a family of transition functions behaving as a set of functionally independent machines. Also, $\wp(\mathbf{M}_{I',E'})$ incorporates all the stages of development that belong $\mathbf{M}_{I,E}$ (the previous development level), since $\mathbf{M}_{I,E}$ is embedded in $\wp(\mathbf{M}_{I',E'})$.

Definition 4.7 gives a formal basis for the definition of the operation of major equilibration. The following definition gives the conditions under which major equilibrations operate: from a bi-structured coherence space $\mathbf{B}$ it is generated the powerset bi-structured coherence space $t\mathbf{B}$, which is regulated with respect to an operation p

Definition 4.8 *Let* $t\mathbf{B} = (t_u\mathbb{B}; t_{\Sigma_I}\Sigma^I_{\mathbb{B}}; t_{\Sigma_E}\Sigma^E_{\mathbb{B}})$ *be the bi-structured coherence space obtained from* $\mathbf{B}$ *by the application of the powerset constructor* $t \equiv (t_u; t_{\Sigma_I}; t_{\Sigma_E})$*, and consider a function* $p : t_u\mathbb{B} \to t_u\mathbb{B}$*. A bi-structured coherence space* $\mathbf{A}$ *is said to be obtained from* $\mathbf{B}$ *by a* major equilibration *regulated by* p*, denoted by* $\mathbf{B} \Rrightarrow_p \mathbf{A}$*, if and only if* $\mathbf{A} = R_p(t\mathbf{B})$*, that is,* $\mathbf{A}$ *is obtained by the regulation of* $t\mathbf{B}$ *with respect to* p*.*

Observe that from Def. 4.8 it follows that $\mathbf{B}$ is embedded in $\mathbf{A}$. Also, $\mathbf{A}$ is a bi-structured coherence space, whose internal and external structures are well defined. Besides, if $\mathbf{B} \Rrightarrow_p \mathbf{A}$ then $p(\mathbb{A}) = p(t_u\mathbb{B})$ is closed for the internal and external functions of $\Sigma^I_{\mathbb{A}}$ and $\Sigma^E_{\mathbb{A}}$, respectively (see [Dimuro 1998]).

In the case of a major equilibration of computing machines, the relevant p operation is $* : \wp(\mathbb{M}_{I,E}) \to \wp(\mathbb{M}_{I,E})$, such that $*[\wp(\mathbb{M}_{I,E})] = \wp(\mathbb{M}_{I,E})_{I',E'}$ (that is, $*$ restricts the universe $\wp(\mathbb{M}_{I,E})$ to its part $\wp(\mathbb{M}_{I,E})_{I',E'}$ that is admissible to both I' and E', where I' and E' are the internal and external factors in the powerset-like domain, generated by the powerset constructor). Then, given a machine $\mathbf{M}_{I,E}$ in the final stage of a certain level of development, the result of its major equilibration is the machine $\mathbf{M}_{I',E'}$ such that $\mathbf{M}_{I,E} \Rrightarrow_* \mathbf{M}_{I',E'}$.

5. Conclusion: On the Future Developments of the Development Model

Further work on the formalization of the notion of developmental computing machines is clearly still necessary. We need to internalize the mechanism of machine development in the developmental machines themselves, by way of some reflection procedure, so that the machine can control itself its own development by controlling the activation of the development factors. We need to make clear the connection of the notion of development to the notion of evolution of populations of computing machines. We need to define the means (relationships and operations) that will allow the developmental machine to interact with the environment, in order to acquire the necessary elements for its development.

6. Post Scriptum: On the Grand Challenge No. 4

The presentation of the Grand Research Challenge No. 4 points out (correctly, it seems to us) that the recent progress in the area of Molecular Biology may inspire new technological developments in CS. However, following the tenets of the classical Theory of Computation, the Challenge states that "the theoretical bases of CS are independent of concrete machines", thus apparently meaning that even the (presently hypothetical) biomolecular computers would fall under the conceptual framework of that theory.

Consistently, thus, the statement of the Grand Challenge No. 4 introduces as issues relevant for that challenge essentially issues of a technical or methodological nature, that is, issues like design techniques, new programming languages, etc.

We remark that going from "silicon", that is, from a technology only able to produce non-developmental machines, to "molecular biology", that is, to a technology able to produce developmental machines, requires much more from CS than mere innovations at the methodological or technical levels.

A technological transition such as that requires a deep epistemological revision of the foundational concepts of the area. Developmental machines can only be adequately coped with at the theoretical level with a conceptual framework where notions of autonomy, growth, development, need, drive, organization, evolution, adaptation, cognition, affect, awareness, will, sociality, values etc., are not thought of *ex post*, but are put right at the beginning at the most fundamental level of its theoretical construction.

Biology should not be taken by CS just as a source of technological resources. It should be taken also as a model of epistemological construction, whose conceptual and theoretical issues should be brought to the domain of computing machines, and suitably instantiated there. The initial work on the identification of organizational mechanisms empowered by the abstract machines of systems biology [Cardelli 2005], for instance, constitutes a beautiful example of the kind of work that may inspire such instantiation.

Acknowledgments. This work is partially supported by CNPq. The authors are very grateful to the referees for their valuable comments, which helped us to improve the paper.

References

Cardelli, L. (2005). Abstract machines of systems biology. *Transactions on Computational Systems Biology III*, LNBI 3737:145–168.

Copeland, B. J., editor (2004). *The Essential Turing – The ideas that gave birth to the computer age*. Oxford University Press.

Costa, A. C. R. (1993). *Machine Intelligence: sketch of a constructivist approach*. PhD thesis, CPGCC/UFRGS, Porto Alegre. (in Portuguese).

Costa, A. C. R. and Dimuro, G. P. (2005). Interactive computation: Stepping stone in the pathway from classical to developmental computation. *ENTCS*, 141(5):5–31.

Demazeau, Y. and Müller, J.-P., editors (1990). *Decentralized Artificial Intelligence*. Elsevier, Amsterdam.

Dimuro, G. P. (1998). *A Global Constructive Representation of Second Order Ordered Systems in Bi-Structured Interval Coherence Spaces, with an application in Interval Mathematics*. PhD thesis, CPGCC/UFRGS, Porto Alegre. (in Portuguese).

Dimuro, G. P., Costa, A. C. R., , and Claudio, D. M. (2000). A coherence space of rational intervals for a construction of IR. *Reliable Computing*, 6(2):139–178.

Dimuro, G. P. and Costa, A. C. R. (2007). Toward a domain-theoretic model of developmental machines. In Kent, F., Löwe, B., and Sorbi, A., editors, *CiE 2007: Computation and Logic in the Real World*, Quaderni del Dipartimento di Scienze Matematiche e Informatiche 'Roberto Magari' of the University of Siena.

Eberhart, R., Shi, Y., and Kennedy, J. (2001). *Swarm Intelligence*. Morgan Kaufmann, Amsterdam.

Escardo, M. H. (1993). On lazy natural numbers with applications to computability theory and functional programming. *SIGACT News*, 24(1):60–67.

Gierz, G., Hofmann, K. H., Keimel, K., Lawson, J. D., Mislove, M., and Scott, D. S. (2003). *Continuous Lattices and Domains*. Cambridge University Press, Cambridge.

Girard, J. Y. (1987). Linear logic. *Theoretical Computer Science*, 50:1–102.

Goldin, D., Smolka, S., and Wegner, P., editors (2006). *Interactive Computation: The New Paradigm*. Springer-Verlag, New York.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall.

Kleene, S. C. (1952). *Introduction to Metamathematics*. D. van Nostrand, New York.

Lewis, L. and Papadimitriou, C. (1997). *Elements of the Theory of Computation*. Prentice-Hall, NJ.

Milner, R. (1989). *Communication and concurrency*. Prentice-Hall, Englewood Cliffs.

Neumann, J. v. (1966). *The Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana.

Newell, A. (1980). Physical symbol systems. *Cognitive Science*, 4:135–183.

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18:87–127.

Newell, A. and Simon, H. (1972). *Human Problem-solving*. Prentice-Hall, E. Cliffs.

Piaget, J. (1971). *Biology and Knowledge*. Edinburgh University Press.

Piaget, J. (1979). *Behaviour and Evolution*. Routledge and Kegan Paul.

Piaget, J. (1980). *Adaptation and Intelligence*. University of Chicago Press.

Piaget, J. (1985). *The Equilibration of Cognitive Structures: The Central Problem of Intellectual Development*. University of Chicago Press.

Piaget, J. (2001). *The Psychology of Intelligence*. Routledge and Kegan Paul.

Rogers, H. (1967). *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York.

Scott, D. S. (1970). Outline of a mathematical theory of computation. Tech. Monogr. PRG-2, Oxford, Oxford Univ. Comp. Lab.

Searle, J. R. (1980). Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3(3):417–424.

Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, 42:230–265.

Wegner, P. (1997). Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80–91.

Wiener, N. (1948). *Cybernetics - or, control and communication in the animal and the machine*. MIT Press, Cambridge. (2nd ed.,1961).