Coupling Computations through Space

Pedro P. B. de Oliveira
INPE-LAC, Caixa Postal 515
Computing and Applied Mathematics Laboratory
National Institute for Space Research
12201-970, São José dos Campos, SP, Brazil
Email: pedrob@lac.inpe.br

Abstract

A conceptual framework for models of coupled computations is developed based upon the assumption that the computations are performed by a population of processing agents whose structure is derived from Turing machines. As a fundamental premise, the agents are embedded in a well-defined space which ultimately provide constraints on the individual movements, thus enabling their autonomous behaviour. The framework takes the form of a taxonomy, according to which Turing-machine components – the tape, the state transition table, or the set of internal states – are let to be shared among the processing agents. Following the presentation of the taxonomy, the $\mathcal{STA}$ coupling model – the one that keeps the internal states and the tape as parts of the processing agents – is picked out and further developed, its features and advantages being stressed in relation to the other models. It is argued that the key aspect of this model is that its notion of computation can only be made sense of at the very interaction between the agents and their environment. A weak and a strong versions of the $\mathcal{STA}$ model are then identified, and an implementation of the latter is briefly discussed. This implementation was performed in an artificial-life system that takes the form of a cellular-automata-based architecture of autonomous agents, which allows to explore the clear-cut notion of space provided by cellular automata. The resulting model – that also embeds into it the features of development and coevolution inherent to the system it is implemented on – is then discussed, in particular by contrasting its features with those possessed by some well-known systems. The aim of the paper is at discussing the issues that its theme gives rise to; thus, no actual computer run derived from a particular set-up of the resulting model is shown.

1 Coupled computations

1.1 Introduction

Complex dynamical behaviours can be obtained out of the coupling of even a small number of iterated systems of a same kind. Various such systems have been reported in the literature, such as coupled maps of various kinds, cellular automata, etc. An interesting class of these systems is the one obtained as a result of coupled computations.

Parallel computations normally refer to the cooperation of computational processes towards the accomplishment of a well-known, predefined task. In coupled computations however, the interest is on the emergence computation that comes out of it.

Depending on the model of computation being used different coupling schemes can be obtained. In particular, depending on the way the components of a model of computation are partitioned, different modes of coupling can be established. For instance, by sharing memory between various von-Neumann machines, a coupling scheme is defined that is distinct from the another built up by sharing, say, one of their internal registers. In the case of Turing machines, the natural partition would be the following set of components: internal states, tape symbols, and state transition table.

In this paper a conceptual framework for models of coupled computations is developed based upon the assumption that the computations are performed by a population of processing agents whose structure is derived from Turing machines (TM). As a fundamental premise, the agents are embedded in a well-defined space which ultimately provide constraints on the individual movements, thus enabling their autonomous behaviour. The framework takes the form of a taxonomy, according to which Turing-machine components are let to be shared among the processing agents. Fol-

0-8186-6715-X/94 $4.00

lowing the presentation of the taxonomy, the *STA* coupling model – the one that keeps the internal states and the tape as parts of the processing agents – is picked out and further developed, its features and advantages being stressed in relation to the other models. It is argued that the key aspect of this model is that its notion of computation can only be made sense of at the very interaction between the agents and their environment. A weak and a strong versions of the *STA* model are then identified, and an implementation of the latter is briefly discussed. This implementation was performed in an artificial-life system that takes the form of a cellular-automata-based architecture of autonomous agents, which allows to explore the clear-cut notion of space provided by cellular automata. The resulting model – that also embeds into it the features of development and coevolution inherent to the system it is implemented on – are then discussed, in particular by contrasting its features with those possessed by some well-known systems. The aim of the paper is at discussing the issues that its theme gives rise to; therefore, no actual computer run derived from a particular set-up of the resulting model will be shown.

1.2 From the Turing gas to Turing machines

One type of coupling scheme is obtained when the unit of coupling is a computable function defined an abstract space. An example of this type is the so-called Turing gas, as defined in [4]. In this system a population of particles are subjected to pairwise collisions with the possibility of formation of new ones which, in turn, enter the chain of the already existing reactions. The Turing gas is a system of coupled computations because the particles are functions coded in a variant of pure-Lisp called *AlChemy* (a shorthand for "Algorithmic Chemistry"), the collisions between the particles being the evaluation of one of the functions having the second as the argument. The Turing gas has been used as a model of systems that have an inherent "constructive dynamics", that is, the ones whose components act on each other constructing new ones which themselves have the ability to take part in the constructive process; a paradigmatic example of this kind of system are the chains of molecular reactions.

Another kind of system of coupled computations is the one based on coupled executions of an assembly-like language that runs in a (typically) virtual machine. [11] and [9] are landmark examples of this kind; [10] is also important, although this work goes beyond the particular coupling scheme currently at focus.[1] It is worth of note the fact that they (indeed, like [4]), tackle the issue of coupled computations without having, however, the need to explicitly recognise it.

[11] features *Tierra*, an artificial life world where a population of programs compete for memory space and CPU time of their host machine. The programs are subjected to an evolutionary process so that the ones that manage to replicate more, get more of these resources, thus guaranteeing their survival. In particular, there are situations in which individuals manage to run instructions that belong to another individual to their own benefit or fate. Therefore, coupled computations in Tierra occur when an area of memory contains instructions that belong to an organism, but are shared by other individuals due to their own individual nature.

The systems *Venus I* and *II*, and *Luna*, discussed in [9] and [10], all implement variants of the idea of a "soup" of instructions spread over an area of memory, where a population of program counters coexist executing the shared code. In terms of coupled computations these systems therefore follow the same approach of Tierra, which is the sharing of instructions among the different computing processes.

A lower-level approach to coupled computations is the one where the unit of coupling are Turing machines. A reference along this line is [8] (assumedly an early inspiration for the Turing gas). This work uses interacting TMs for studying functional self-organisation, and is based on a binary string encoding of their transition tables.

Recently Rucker (1993) released a software package and accompanying book which, in spite of simply aiming at being an intertainment artificial-life system, provides an interesting example of coupled computations. This alife world is inhabitted by a population of two-dimensional Turing machines whose individual transition tables are coded in the organisms' genotypes. The organisms' environment is seen as a two-dimensional tape that is shared among all individuals. As they move about they leave trails that can be followed by the others. The symbols that make up the trails are, therefore, the symbols that are written on the tape which, when read by other organisms, provide the efective coupling among the computations individually defined in each Turing machine.

[1] Its theme is the more encompassing concept of dynamics of "self-programmable matter"; although not explicitly recognised in the paper, it essentially corresponds to the notion of constructive dynamics defined in [4].

1.3 The role of space

The implications of different coupling schemes can be seen from various perspectives, such as, the degree of perturbation one machine has onto another (for instance, sharing a register versus sharing the entire memory space); or the adequacy of the scheme, when it is considered as a model of some phenomenum (the model of computation used in the Turing gas fits nicely into the analogy of the Lisp-functions being the particles or objects of the world, and the capability of evaluating the functions being the underlying physics). Another perspective is the role of the space in which the coupling takes place.

Explicitly or not, some notion of space is embedded in any coupled computation scheme. For instance, the notion of space used in systems that rely on coupled executions of assembly-like languages, like Tierra and the Venus family, is defined by the memory space of the computer involved. The space inhabitted by the organisms in Rucker's systems is a lattice on the surface of a torus. Even in the context of other applications rather than coupled computations, a number of systems have used well-defined notions of space, such as in neural netwoks, genetic algorithms and robotics. In contrast, although the activity in the Turing gas has been metaphorically described as taking place in a "volume", this is in fact an abstract, rather ill-defined space.

The crucial point about space is that it can be used as a very natural way to integrate the coupled computations, the upshot of it being a substantial gain in autonomy for the process. This is the alley to be explored herein. By autonomy we mean the lack of centrallised control, an intrinsic parallellism, and the fact that the agents involved have an individual ability to act.

2 Coupling Turing machines through space

2.1 Assumptions and definitions

In this section and the next, models for coupling Turing-machine-based computations will be discussed. All these models are based on a population of TMs such that:

- They are embedded in a space, thus becoming possible to distinguish a TM from its *environment*, that is, the rest of the space, apart from them.
- They have the autonomy to move in the space.
- Most of the environment is free for the TMs to move through.
- There is a special part of the environment, denoted by ***interaction site***, that can be reached by the TMs but not traversed by them.
- The only direct interaction between the TMs is one obstructing the way of another as they move.

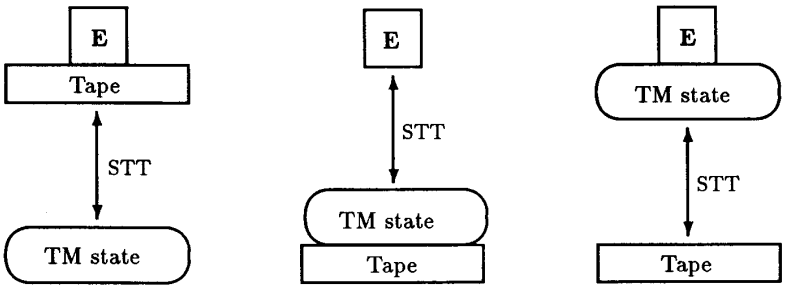
Let us consider the situations in which any of the main components of a TM (tape, state transition table and internal state) is taken out of each member of the population, and ascribed to the interaction site. This situation yields a coupling scheme between all TMs in the population, insofar as any computation that is performed at them depends on the content of the interaction site, which is shared among all the machines. Let us denote the shared component as the ***coupling unit***.

As a consequence of "disabling" a TM as above, the entities that are formed by the remaining components can no longer be characterised as complete Turing machines; let us think of these moving entities – that can take part in a computation defined in terms of Turing machines – as ***agents***.

2.2 Models of coupling

Although the action of a state transition table only makes sense, by definition, at the very interaction between tape and states, in a coupling scheme the table could well be confined to the environment, to the agent, or to both at the same time (in which case it can be thought of as being a part of the entire space where environment and agents are defined in). In [12], for instance, the state transition table of the Turing machines are internally defined in the organisms as a code in their genotypes (while the two-dimensional tape is the entire environment inhabitted by the organisms).

Depending on the choice of which one of the three components of a Turing machine is let to be shared among the others, nine distinct coupling models can be obtained. Figure 1 features three of them; the others are irrelevant for present purposes but can be envisaged by considering the state transition table either in the environment or in the agent. In the models shown the state transition table plays the role of the "physics" of the world that creates the condition for a step of computation to be performed involving an agent. In other words, the state transitions can be



$\mathcal{SoA}$ Model ($\mathcal{S}$tate-only in the $\mathcal{A}$gent)

$\mathcal{STA}$ Model ($\mathcal{S}$tate and $\mathcal{T}$ape in the $\mathcal{A}$gent)

$\mathcal{ToA}$ Model ($\mathcal{T}$ape-only in the $\mathcal{A}$gent)

Figure 1: The three models of coupled Turing machines, in which the state transition table (STT) is part of both the environment and the agent; that is, it is part of the space they are defined in. The agents are represented at the bottom and the interaction sites at the top.

thought of as belonging to both the agent and the environment. The unit of coupling resides in the interaction sites. One possibility is the interaction site representing either the tape or the state of the TM; another is to keep the tape and state as part of the same agent, while allowing the interaction site to act as the "physical support" for the computations to be performed. These three models of coupling are named, respectively, $\mathcal{ToA}$ (tape-only in the agent), $\mathcal{SoA}$ (state-only in the agent) and $\mathcal{STA}$ (state and tape in the agent).

The essence of all models of interest here is that the coupling will happen according to the coordinated movement of the population of agents in the space they are defined in. As they move about, they interfere with each other's trajectories, then leading different agents to different interaction sites at different times, in a totally autonomous and decentralised fashion.

Because the $\mathcal{ToA}$ and $\mathcal{SoA}$ models fully share so fundamental parts of the structure of a TM – its internal state and the tape configuration – two major problems arise:

- Both models become too brittle in terms of their ability to support coupled computations in any practical way. The outcome of the couplings would too often lead to meaningless computations, much like the effect of arbitrarily putting together pieces of code from a standard programming language.
- The process of identification of the end of a computation becomes irremediably impaired. The end of a computation requires that not only a final state is reached, but also that the TM head points at a predefined symbol at the tape. Because the internal state and the tape are always disconnected in both $\mathcal{ToA}$ and $\mathcal{SoA}$, there is no straightforward way to identify when a computation has been completed. The only possibility is by fully inspecting the state of the world at each iteration; but this is a trivially uninteresting situation.

So far we have considered the interaction site as a singleton. This has been done because it is easier to convey the idea of coupling with an unique coupling unit. However, multiple interaction sites could alternatively be used. The major consequence is that the resulting coupling scheme would be even tighter, insofar as the interference possibilities between agents and sites would certainly increase. Naturally, the problems mentioned above for the $\mathcal{ToA}$ and $\mathcal{SoA}$ models would become even more critical.

3 The $\mathcal{STA}$ model

Because of the problems aforementioned a case will be made in this section for the advantages of using the $\mathcal{STA}$ model. First of all, let us assume that multiple interaction sites are in use.

The role of the interaction sites is really twofold: they are an essential part of the computation (for instance, by being the repository of the tape in the $\mathcal{ToA}$

model), and they provide a spatial reference for when the coupling should effectively take place.

While in $\mathcal{ToA}$ and $\mathcal{SoA}$ the interaction sites already possess one of the defining components of the TMs, in $\mathcal{STA}$ they do not, since tape and TM state are confined to the agents. From this observation it is clear that the coupling in $\mathcal{STA}$ is not as as tight as it is in the other two. Brittleness therefore has decreased, thus leaving room for multiple interaction sites to be used. It is worth noting that this scheme is indeed more appealing than a single interaction site, insofar as it explores the parallel nature of the various coupled computations.

Another consequence is that the problem mentioned above of the identification of the end of a computation is now solved. Whenever the individual has reached a final state and the head of the corresponding TM is pointing at the last symbol it should, the computation has finished and the corresponding tape represents its outcome.

Finally, the $\mathcal{STA}$ model *necessarily* leads to a modified Turing machine where, in addition to the standard TM components, the new dependency on the *state of the interaction sites* would have to be made explicit.[2] In fact, two (equivalent) possibilities for creating the new state transition table would be: the addition of the site state as a new entry variable, thus yielding a three-dimensional table; or the modification of the original table, by replacing the TM state entry by a new entry formed by pairs of the two kinds of states. The model thus has a "hint" of the $\mathcal{ToA}$ model, in the sense that the coupling scheme works as though the interaction site would be the repository of a new state which the steps of computation become dependent upon.

Depending on the extent of these table modifications, which are related to the degree of coupling that is allowed for the interaction site, two distinct possibilities for the $\mathcal{STA}$ model can be distinguished: the weak and the strong versions.

3.1 The weak $\mathcal{STA}$ model: only the form of the table is modified

In the weak $\mathcal{STA}$ model the role of the interaction sites is to enable or not a computation step. That is, some sites would allow the step of computation to normally occur, as defined in the state transition table, and some would prevent it, leaving the state-tape configuration unchanged at that point. At the same time, the state of the interaction site might be modified, according to the agents' configuration. But no new entries in the table would be created, keeping its original content the same. They would only have to be modified to reflect the new table format.

Hence, coupling in the weak $\mathcal{STA}$ refers only to the fact of whether a step of computation will be able to be performed as the result of an interaction. What is at stake is the speed at which the (entire) computation will be performed, or whether it will be completed at all (not getting stuck at some entry of the transition table). For different runs, any computation that starts with the same initial state-tape configuration will lead to a final configuration that will be the same in all runs. In other words, all computational pathways (sequences of steps of computation as defined by the entries in the state transition table) are unique, regardless of their being related to a correct computation or not. Naturally, this behaviour is nothing more than the one normally expected from functioning programs written in standard programming languages.

By contrasting this weak version of the $\mathcal{STA}$ model with $\mathcal{SoA}$ and $\mathcal{ToA}$ models, it is clear that while the coupling provided by the latter two is too tight (too much coupling), in the weak $\mathcal{STA}$ scheme it is too loose (too little coupling). So, while the interaction sites in $\mathcal{ToA}$ and $\mathcal{SoA}$ are *primary agents* of the coupling, insofar as they incorporate fundamental components of the computations, the role of the interaction site in the weak $\mathcal{STA}$ is simply one of *enabler* of a computation step. The desired degree of coupling should be somewhere between the two extremes; one that would allow the computation steps to become dependent, in a stronger way, on the state of the interaction sites.

3.2 The strong $\mathcal{STA}$ model: the table content is modified

This stronger dependence means that the result of an interaction should be expressed not only in terms of the corresponding step of computation being able to be performed or not, but also in terms of which one it will actually be. That is, new entries should be created in the state transition table with new actions corresponding to the new, possible interactions.

With the modification of the transition table (for example in either of the ways suggested earlier in this section), a coupling scheme is achieved that, depending on the sequence of interaction sites the agent comes across, the agent may be led into a distinct sequence of computational steps, that is, into distinct computational pathways. The major consequence is that, for different runs, the same initial state-tape configuration

[2] Naturally, it is possible to think of table modifications also in $\mathcal{SoA}$ and $\mathcal{ToA}$; but while this would be a deliberate action in them, it is a necessity in $\mathcal{STA}$.

may lead to distinct final configurations. And finally, no computational pathway is uniquely determined by its corresponding initial configuration.

Assuming the table has been modified, there is no unique way to traverse it. Naturally, the model presupposes that it will take place through the coordinated movement of the population of agents.

So, even though an experiment can be run with a single state transition table various distinct functions can be identified. How the transition table should be modified is a matter of implementation, the possibilities being the composition of state transition tables from distinct, well-formed functions; the addition of state transitions that do not necessarily constitute a full function; or the mixing-up of state transitions from whatever origin. The new table formed as above then has the potential to yield not only the original functions that might have been used, but also others that are the result of "interferences" between the individual contents of each one of the primitive tables, or the individually added state transitions.

Summing up, what we have gained with the strong $\mathcal{STA}$ model is a tighter model of coupling than the weak version, one that opens up the possibility of distinct functions to emerge. But at the same time, the coupling is loose enough to provide us with a way to identify the end of a computation.

4 Embedding the $\mathcal{STA}$ model in an artificial-life system

In this section we hint at the implementation of the strong version of the $\mathcal{STA}$ model within an artificial-life system, and discuss what is gained out of this mix in terms of a model of coupled computations that incorporates the aspects of coevolution and development of the artificial-life system. No actual computer run derived from a particular set-up of the resulting model will be shown; only the conceptual issues that emerge will be of interest for present purposes. An in-depth analysis of the results associated with a particular set-up of coupled computations will be discussed elsewhere.

4.1 Background: Enact, an artificial-life world

The implementation was performed within Enact, a family of two-dimensional cellular automata whose temporal evolution on a periodic background can be described in terms of the metaphor of an artificial-life world where a population worm-like agents undergo a coevolutionary process. During their lifetime, the agents roam around, sexually reproducing, interacting with the environment, and being subjected to a developmental process.

Whenever possible Enact's agents move; by design, they move either leftwards or diagonally, and never "bump" into each other. Agents can only "touch" environmental configurations that play the role of interaction sites. By designing specific state transitions to be active only in these sites, it is possible to have specific, controlled interactions taking place between environment and agents. Any agent then interacts with any other in the world through the environment. Since each agent's individual movement may influence the way the other agents move, their developments have a major interdependence through movement. Many additional details about Enact can be found in [1] and [2].

Enact is a system that embeds the models of computation discussed here, in particular the $\mathcal{STA}$ model, in either of its versions; for instance, in [2] an implementation the $\mathcal{SoA}$ model was showed, and in [3] we showed within Enact the implementation of a population of Turing machines that would be able to act along the lines of the strong version of the $\mathcal{STA}$ model. Its main feature is that it casts the issue of coupled computations in terms of an artificial-life world. Most importantly, it is the lifetime history of coupled movement of the population that determines what an individual agent will develop into. And it is an agent's development that is interpreted as a function, the final state of the agent being the outcome of the function.

4.2 Enact's approach to coupled computations in perspective

Although recognising cellular automata as providing

> "...a powerful approach to the study of the emergence of loops between objects and functions ...",

Fontana ([4, page 198]) then remarks that it

> "...becomes, however, difficult to study the consequences of such a loop at the same level of description that has been used to study its emergence."

A related concern is expressed in [10, page 219] when discussing cellular automata as "self-programmable" systems. In that paper it is stated that the

> "...main difficulty with the CA approach seems to be associated with ... the extreme low-level representation of interactions."

It should be noted that the approach used in Enact provided an effective way to solve the worries above, insofar as the use of a population of autonomous agents – the processing units involved in the computations – are realised at a higher level than the one Enact itself is implemented at. That is, while Enact is defined from basic state transitions, the population of processing agents is mainly defined through the high-level concepts of the system (such as agents, phenotypes, memetypes and so on.)

Several issues can be explored in a comparison between the $\mathcal{STA}$ model of coupled computations in Enact and other approaches. What follows is an attempt to compare some of the aspects, mainly with respect to the Turing gas and Tierra. The model just discussed has the following features:

- *Evolutionary capabilities.* Just like Tierra, the model can be used within an evolutionary context, even though the actual evolutionary possibilities is not the same for each of them. The Turing gas however lacks this feature, which is even explicitly recognised in [4].

- *Focussed emergent computations.* The model can be used to tackle the problem of emergent computations (or, in particular, of emergent functions) even in small regions of the function space. The point is that the function space implicitly defined by the state transition table – that characterises the interactions between the agents and the environment – can be controlled in an independent fashion. This is possibly the most fundamental aspect of the system. The tractability that is gained implies that it becomes possible to approach the issue of functional emergence by looking at the *actual* functions that emerge. In fact, we have already performed extensive analyses of a set-up which enables the emergence of a wealth of functions that operate over a sequence of numbers, such as sorting them in increasing order; these results will be reported elsewhere.

- *Ability to link functional emergence to such the (apparently) disconnected concept as "phase" transition.* By enabling the process of functional emergence to be focussed in a region of the function space, it becomes possible to create a link between the issues of functional self-organisation and phase transitions in some dynamical spaces (see [6]).[3] For instance, it becomes possible to refer to criticallity phenomema by means of the situations in an agent's lifetime which are determinant of its long term development. That is, the critical points that determine which computable function the development of an agent will end up being characterised by. Aspects of the reversibility of computations also come up in this context. These aspects have also been addressed in the set-up mentioned above

- *Copying or reproductive function is not essential.* A copy or reproduction function does not play any major role in Enact as they do in various of the experiments discussed for the Turing gas, or in virtually all reported experiments performed with Tierra. Indeed, all insteresting reported outcomes from the latter system depend on the existence of the so-called "Ancestor", a self-reproductive function that is innoculated in the Tierran soup at the start of a run. A step towards an exception was reported in [13]; with the addition of a new register to Tierra, selective pressure was allowed according to the processing of the content of the register. By acting as a connection of Tierra with the outside world, the added feature provides an additional way to drive Tierra's evolution beyond mere reproduction.

- *Functions with any number of parameters.* For the purposes of making AlChemy's implementation easier, it can only handle functions of a single parameter. Naturally, this is a strong constraint that restricts the sorts of emergence that can be observed; Fontana (1992) himself recognises the problem. But again, it should be clear that the problem is not a consequence of the model, but only of its implementation.

- *Robustness.* The major problem when computer programs become the subject of evolutionary and of self-organisation processes is how to achieve robustness, i.e., how to escape from the brittleness of their semantics when arbitrarily putting chunks of code together. One way or the other this problem has been solved in AlChemy, Tierra and various other systems. Turing machines have a very robust semantics because they simply handle states, which are indifferentiable from each

[3] The primary concern on phase transitions presented in [6] was its characterisation in the rule space of cellular automata. The fact that Enact is implemented as cellular automata is even more appealing with this respect.

other. It is such a robustness that enables the approach supported by the $\mathcal{STA}$ model. No matter how strange it may sound in principle, it is worth remarking that the set-up mentioned above did not use Turing machines. But since the functions it dealt with only required pure number values, the argument of robustness still holds in that case.

- *Autonomy.* It is the movement of the agents that determine the outcome of the functional emergence; but movement is part of the nature of the agents, constrained by the availability of free space. Hence, there is no need for any sort of centralised process that would be in charge of determining which components of the system would have to be used at a certain moment. The latter is exactly what happens in the Turing gas in regard to the need of arbitrating the pairs of Lisp-particles that will collide at a given instant. In Enact, this "decision" is not only decentralised, but also is just a natural consequence of the dynamics of the system.

5 Final remark: towards a model of computation for natural phenomena

The computer metaphor has been widely used to describe natural phenomena; its success, however, is questionable. For instance, as [14] reminds us, two very common misconceptions can be perceived: in cognitive science, when considering environment as data that is given to a program in the cognizer; and in biology, when assuming the genome of an organism as a program that is run by the biomolecular machinery. I believe that the main cause of these flaws is not on the approach itself. Instead, it comes from the model of computation that is used to ground the metaphor. It is expected that the issues raised in the context of the $\mathcal{STA}$ model of computation may shed light on the track that leads to descriptions of natural phenomena in harmony with the use of the computer metaphor.

Accordingly, the fact that the state transition table of the models discussed herein were let to be shared by agents *and* interaction sites is meant to provide a notion of computation that has to be regarded as taking place only at the very interaction between agents and environment.

In the strong $\mathcal{STA}$ model the role of the environment has been lessened, if compared to the $\mathcal{ToA}$ and $\mathcal{SoA}$ models, and has been strengthened in relation to the weak $\mathcal{STA}$. It still has an active role in the computation, but became a *mediator* rather than an enabler that it is in the latter model, or a primary agent of the computation that it is in the former two. The environment's role has become the provision of *active physical support* for the computation to occur.

In addition, the fact that the actual functions that are computed at the agents critically depend on their lifetime history of interactions, strengthens the role of the interactions, and consequently, the role of the agents, since they are major responsibles for their movement. But since the agent itself provides the place where the outcome of the computation is visualized, it is tempting to say that in the strong $\mathcal{STA}$ model the agent has become the *subject* and the *object* of the computation, a notion that is based on an idea originally developed in [7] in the context of biological evolution.

Other aspects of the $\mathcal{STA}$ model of coupled computation which are worth bearing in mind include its intrinsic parallellism, and the stress on the notion of computation as a dynamical process. Even more fundamentally, the definition itself of the coupling scheme, as well as its reliance on the notion of autonomy of the processing agents could only be achieved by explicitly resorting to a well-defined notion of *space* that permeates all activity.

Acknowledgements

Early discussions with Phil Husbands and Inman Harvey helped me shape up the ideas that led to the concepts expressed herein. This work was partially done in England, thanks for the following Brazilian institutions: CNPQ (National Council for Scientific and Technological Development) and INPE (National Institute for Space Research).

References

[1] Pedro P. B. de Oliveira. "Enact: An artificial-life world in a family of cellular automata." CSRP-248/92, School of Cognitive and Computing Sciences, University of Sussex, Brighton, UK, Sept. 1992.

[2] Pedro P. B. de Oliveira. "Methodological issues within a framework to support a class of artificial-life worlds in cellular automata". In: D. G. Green & T. Bossomaier, editors. *Complex Systems: From Biology to Computation.* IOS, Amsterdam, 82–96, 1993.

[3] Pedro P. B. de Oliveira. "Collapsing a Coevolutionary Process into a Computable Function".

Submitted to: D. B. Fogel & W. Atmar, editors. *Evolutionary Computation: From Biological Foundations to Intelligent Systems.* Kluwer, 1994.

[4] Walter Fontana. "Algorithmic Chemistry". In: J. D. Farmer, C. Langton, S. Rasmussen & C. Taylor, editors. *Artificial Life II*, Addison-Wesley, 159–209, 1992.

[5] J. E. Hopcroft & J. D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 1979.

[6] C. G. Langton. "Computation at the edge of chaos: Phase transitions and emergent computation". *Physica D*, 42(1–3):12–37, 1990.

[7] R. C. Lewontin. "The Organism as the Subject and Object of Evolution". *Scientia*, 118:63–82, 1983.

[8] John S. McCaskill. "Polymer Chemistry on Tape: A Computational Model for Emergent Dynamics". Preprint from the Max-Planck Institut für Biophysikaliche Chemie, Göttingen, Germany, 1989.

[9] S. Rasmussen; C. Knudsen; R. Feldberg & M. Hindsholm. "The Coreworld: Emergence and Evolution of Cooperative Structures in a Computational Chemistry", *Physica-D*, 42:111–134, 1990.

[10] S. Rasmussen; C. Knudsen & R. Feldberg. "Dynamics of Programmable Matter". In: J. D. Farmer, C. Langton, S. Rasmussen & C. Taylor, editors. *Artificial Life II*, Addison-Wesley, 211–254, 1992.

[11] Thomas S. Ray. "An approach to the synthesis of life". In: J. D. Farmer, C. Langton, S. Rasmussen & C. Taylor, editors. *Artificial Life II*, Addison-Wesley, 371–408, 1992.

[12] Rudy Rucker. *Artificial Life Lab*, Waite Group, Corte Madera, CA, USA, 1993.

[13] Walter A. Tackett. "Fitness and Adaptation of Digital Organisms". Talk given at *Artificial Life III*, Santa Fe, NM, USA, 1992.

[14] F. Varela; E. Thompson, & E. Rosch. *The Embodied Mind: Cognitive Science and Human Experience.* MIT Press: Cambridge, MA, 1991.