Self-Programming and Correcting Languages

We call self-programming the worldwide procedure that animates computational operationally autonomous systems, i.e. the application of mutually operational and semantic conclusions. Therefore, a self-programming machine (the self) is instituted by three categories of code:

C1: The programs that perform on the word and the self

These are the programs that calculate the assembly and implementation of code. They function in any of the three categories subsequently.

C2: These are the models that designate the program in C1, comprising the self in the world and programs in the self.

C3: The inputs/outputs of the machine, past, present and anticipated states of the self are considered in C3.

In the non-appearance of values, for impulsive origin we have to accept the presence of a set of primary hand-crafted knowledge. It contains the ontologies, states, models, internal drives, exemplary behaviors and programming skills. Self-programming needs an innovative course of programming language containing low complexity, high expressivity and runtime reflectivity. Consuming any of the typical languages accessible today to write a program that makes additional one and assimilates it in an existing system is an actual task.

To infer the determination of source code, a program would have to appraise the association code against a proper model of the machine. The language assemblies are not replicated and it is virtually unmanageable from the sole reading of the memory to restructure objects. What is good for a human programmer is not so good for a system to create its own code in real-time. Wonders are striking forms revealing a primary and feasibly concealed process. They must be taken hypothetically at any scale e.g. from the scale of enhancing some low-level programs to the scale of reconfiguring the whole system.

Consequently, we describe states as global and stable systems of action at the atomic level. This states the stable actuality of specific programs and objects (models, inputs/outputs, etc.) while higher-level states are intangible processes that synchronize in the state space identify or control the completion of the programs that produce these states. From this outlook, making sense is recognizing - or infuriating fundamental creation relationships between processes: an occurrence makes logic through the progress of its pragmatics, the properties it aggravates, in the system, and P means another phenomenon. Making sense is a procedure achieved irrespective of the length or duration of production graphs; it is a process that can be perceived or adopted at any subjective scale.

In the sphere of computational systems, autonomy can be operationalized by the notion of self-programming for which we have offered a prototype: a dynamic, real-time and self-referential construction for building and grounding information and procedures in high-

dimensional topological spaces. For such systems, binding the rise of high-order groups in a commonaccessible way calls for innovative engineering practices.