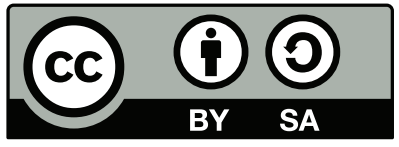
Laboratory of Computer Science Education

Notional machine

Alberto Montresor

Università di Trento

2021/05/19



Notional machine

The term notional machine was introduced to Computing Education Research by Benedict du Boulay, who used it to refer to "the general properties of the machine that one is learning to control" as one learns programming [du Boulay 1986].

A notional machine is an idealized computer "whose properties are implied by the constructs in the programming language employed" [du Boulay 1986], but which can also be made explicit in teaching [du Boulay et al. 1981].

Notional machine

Abstractions are formed for a purpose; the purpose of a notional machine is to explain program execution.

A notional machine is a characterization of the computer in its role as executor of programs in a particolar language or a set of related languages.

Abstract but sufficiently detailed...

Compare with Operational semantics:
`https://en.wikipedia.org/wiki/Operational_semantics`

Notional machine

The machine (or system) T we were programming (and which we wanted the students to understand), was not really a computer, at least in the classic, hardware, sense.

In the case of C++ and Java languages, T is an abstraction combining aspects of the computer, the compiler and the memory management scheme. Our T is not nearly as "knowable" [as some other systems]. That does not relieve us of the responsibility of at least trying to define it.

Aggiungo io: nell'evolvere dell'esperienza nella programmazione, nella macchina T bisogna aggiungere anche le librerie

Notional machine: Cosa è?

- is an idealized abstraction of computer hardware and other aspects of the runtime environment of programs;
- serves the purpose of understanding what happens during program execution;
- is associated with one or more programming paradigms or languages, and possibly with a particular programming environment;
- enables the semantics of program code written in those paradigms or languages (or subsets thereof) to be described;
- gives a particular perspective to the execution of programs; and
- correctly reflects what programs do when executed.

Notional machine: Cosa non è?

- A notional machine is not a mental representation that a student has of the computer, that is, someone's notion of the machine. Students do form mental models of notional machines, however;
- A notional machine is not a description or visualization of the computer either, although descriptions and visualizations of a notional machine can be created by teachers for students, for instance;
- Finally, a notional machine is often not a general, language- and paradigm- independent abstraction of the computer

Discussione

- Mental model
-

Mental model

A mental model is a mental structure that represents some aspect of one's environment

Norman [1983] suggested that people rely on their mental models to develop behavior patterns that make them feel more secure about how they interact with systems, even when they know what they are doing is not necessary.

Mental models are often not the product of deliberate reasoning; they can be formed intuitively and quite unconsciously

Mental model

- reflect people's beliefs about the systems they use and about their own limitations and include statements about the degree of uncertainty people feel about different aspects of their knowledge;
- provide parsimonious, simplified explanations of complex phenomena;
- often contain only incomplete, partial descriptions of operations, and may contain huge areas of uncertainty;
- are "unscientific" and imprecise, and often based on guesswork and naive assumptions and beliefs, as well as "superstitious" rules that "seem to work" even if they make no sense;
- are commonly deficient in a number of ways, perhaps including contradictory, erroneous, and unnecessary concepts;

Mental model

- lack firm boundaries so that it may be unclear to the person exactly what aspects or parts of a system their model covers – even in cases where the model is complete andcorrect;
- evolve over time as people interact with systems and modify their models to get workable results;
- are liable to change at any time; and
- can be "run" to mentally simulate and predict system behavior, although people's ability to run models is limited.

Mental models are required

A mental model of a notional machine allows a programmer to make inferences about program behavior and to envision future changes to programs they are writing.

It is widely accepted that programming requires having access to some sort of 'mental model' of the system [Canas et al. 1994]

Most mental models are not adequate

> attribute students' fragile knowledge of programming in considerable part to a lack of a mental model of the computer [Perkins et al. 1990]

> [students'] mental model of how the computer works is inadequate" [Smith and Webb 1995]

Mental models: intervention

A challenge of programming education is to facilitate the evolution of students' models so that they have these features [Stability, accuracy, generality]. Teaching about an explicit notional machine may decrease the level of freedom that learners allow themselves as they form mental models and may result in better models

Aiding mental model formation as early as possible is important, as changing an ingrained but flawed mental model is more difficult than helping a model to be constructed in the first place.

Mentals models and tracing

Tracing is a key programming skill that expert programmers routinely use during both design and comprehension tasks [Adelson and Soloway 1985; Soloway 1986; Detienne and Soloway 1990].

Perkins et al. [1986] found that many novices do not even try to trace the programs they write, even when they need to in order to progress. In their study, "students seldom tracked their programs without prompting."

Why?

- a failure to realize the importance of tracing,
- a lack of belief in one's tracing ability,
- a lack of understanding of the programming language,
- or a focus on program output rather than on what goes on inside.

Status representation

A status representation of a complex program involves an amount of information that often exceeds the capacity of working memory, which is why we use external aids such as scraps of paper and debugging software.

Novices need concrete tracing often, but are not experienced at selecting the right "moving parts" to keep track of in the status representation, causing them to fail as a result of excessive cognitive load

Constructivism in the Notional Machine Debate

Greening: Multiple Perspectives

vs

Ben-Ari: Prior Knowledge of the Computer

Program Dynamics as a Threshold Concept

An argument has been presented that program dynamics constitutes a major transformative threshold that beginner programmers must cross.

The ability to view programs as dynamic is required to genuinely understand a legion of other concepts and distinctions: variables and values; function declarations versus function calls; classes, objects, and instantiation; expressions and evaluation; static type declarations versus execution-time types; scope versus lifetime; and so on. The dynamic use of memory to keep track of program state is central to much of this integrative power

Program Dynamics as a Threshold Concept

Irreversibility: However, at least the present author has never heard of a programmer forgetting how to see programs as dynamic, traceable entities once they have made that concept their own, nor does he expect to hear of one

The Notional Machine as a Learning Objective

- The Notional Machine as a Learning Objective
  - Transforming learners' perspective
  - Teaching about a machine model
- Techniques for teaching
  - Pedagogy for Threshold concepts
  - Program visualization
  - The impact of learning activities
  - The impact of curricular ordering

Altri spunti interessanti

"To understand a program you must become both the machine and the pro- gram" [Perlis 1982].

Where findings from different research traditions point in the same direction, they strengthen each other. Where they point in different directions, they give us food for thought and remind us that learning is complex and multilayered.