## What is a process?

Robin Milner, September 2009

The concept of process has become increasingly important in computer science in the last three decades and more. Yet we still don't agree on what a process is. We probably agree that it should be an equivalence class of interactive agents, perhaps concurrent, perhaps non-deterministic. Whatever they are, processes play different roles; these range from specifying the interactive programs that we design, to discrete modelling of naturally occurring behaviour (as in biology). In between lie systems that we partially design but are embedded in a natural environment (e.g. ubiquitous systems) or in environments such as the internet or washing machines or cars, designed by engineers of varying disciplines.

Thus informatic processes have become increasingly relevant outside computer science as it was three decades ago, when both CSP and CCS were first launched. In the preface to my book Communication and Concurrency (1989) I said

> *If readers will apply it [CCS] to their own problems in designing concurrent systems, or use it to clarify their conceptual grasp of existing systems (man-made or otherwise), then we can learn from their experience more about the pertinent concepts for concurrency and communication and how they should be managed.*

So I would like to revisit CCS in the light of the wider experience of three decades. Before that, it is wise to clear up an ambiguity in the way the word 'process' is used; it sometimes means an equivalence class of systems or agents, and other times it is used to mean a system or agent that belongs to such a class. I shall use it in only in the former sense.

The principles on which the CCS notion of process is based appear not only sound, but also relevant to the wider range of informatic systems that occur naturally or are only partly designed. There are two main principles; informally, they are as follows:

> **Preservation by context:** It is generally accepted that a process, i.e. an equivalence class of agents, should be preserved by every context, i.e. should be a congruence. This is generally understood to refer to *static* contexts, i.e. the operations by which a larger agent is built from smaller ones. But agents are inherently dynamic, so it is natural to demand that the equivalence is preserved also by *dynamic* context, i.e. by any interaction with the environment.
>
> **Instantaneous interaction:** An interaction between two processes, or between a process and its environment, is an action synchronized between the parties (this is easily generalised to multi-party interactions.) It is not buffered, because buffers can themselves be defined as processes. Each party can infer no more from the interaction than that it has occurred.

The first principle means that one can talk of a *process* $p$ (not merely an agent) reaching a new state $p'$, via interaction with its environment. For we know that if any agent $a$ in the class $p$ reaches the state $a' \in p'$ by this interaction, then any other member of $p$ can also reach a state in $p'$. This,

of course, is the essence of bisimilarity. So we are saying that bisimilarity arises as soon as we require equivalence to be preserved by dynamic as well as static contexts.

This pair of principles has been carried forward in all the subsequent work I have done on processes. It contradicts the remark of Nain and Vardi in the last paragraph of their paper suggesting process theory has lacked guiding principles. They may not like the above principles, but that is another matter!

For the present, let me recall one of the earliest views on process theory. Carl-Adam Petri said or wrote the following long ago (I cannot find the article or remember the exact wording):

*Information enters a non-deterministic process in finite quantities throughout time.*

This must be admitted by anyone who does not believe that we can incorporate within a system all the hidden variables that determine its behaviour.

Consider, for example, the activity in a medical system in which sensors on or within your body monitor its state and transmit information to the hospital, where a computers or other humans respond with guidance or even with instructions to activators within your body. If we want a model of the informatic component in that system, together with its interaction with sensors and actuators, than we can't avoid considering in which states, and in what ways, the non-determinism is resolved. To put this more technically: We shall not be able to avoid model-checking the system for non-trivial properties expressible in CTL but not in LTL. Recall that bisimilarity corresponds to satisfying the same sentences in quite a large subclass of CTL formulae.

I cannot conceive how a model that forgets when and how non-determinism is resolved will be acceptable for modelling such real informatic processes. Yet we are faced with modelling informatic phenomena (designed or naturally occurring) in the real world that can stand well beside the models of other scientists.

I am grateful to Moshe Vardi for proposing a continued debate on this topic. To me, it's one of the most important issues in Computer Science. I have not said here everything I would like, but perhaps enough to encourage others to join in.