Cognitive Science Research Papers: No 13. (Sussex University, 1981)
(Slightly shorter version appeared in Proceedings Cognitive Science Conference 1981)
Slightly re-formatted 23 Feb 2019

SKILLS, LEARNING AND PARALLELISM

Aaron Sloman,
Cognitive Studies Programme,
University of Sussex,
Now School of Computer Science, University of Birmingham
http://www.cs.bham.ac.uk/~axs/

The distinction between compiled and interpreted programs plays an important role in computer science and may be essential for understanding intelligent systems. For instance programs in a high-level language tend to have a much clearer structure than the machine code compiled equivalent, and are therefore more easily synthesised, debugged and modified. Interpreted languages make it unnecessary to have both representations. Further, if the interpreter is itself an interpreted program it can be modified during the course of execution, for instance to enhance the semantics of the language it is interpreting, and different interpreters may be used with the same program, for different purposes: e.g. an interpreter running the program in 'careful mode' would make use of comments ignored by an interpreter running the program at maximum speed (Sussman 1975). (The possibility of changing interpreters vitiates many of the arguments in Fodor (1975) which assume that all programs are compiled into a low level machine code, whose interpreter never changes).

People who learn about the compiled/interpreted distinction frequently re-invent the idea that the development of skills in human beings may be a process in which programs are first synthesised in an interpreted language, then later translated into a compiled form. The latter is thought to explain many features of skilled performance, for instance, the speed, the difficulty of monitoring individual steps, the difficulty of interrupting, starting or resuming execution at arbitrary desired locations, the difficulty of modifying a skill, the fact that performance is often unconscious after the skill has been developed, and so on. On this model, the old jokes about centipedes being unable to walk, or birds to fly, if they think about how they do it, might be related to the impossibility of using the original interpreter after a program has been compiled into a lower level language.

Despite the attractions of this theory I suspect that a different model is required. In chapter 8 of Sloman (1978) I drew attention to familiar facts about children counting which suggest that instead of using a single program interleaving the production of a new numeral and pointing at a new object, they run two processes in parallel, using a third process to monitor them and keep them in step, or abort them if they get too far out of step. If children used a single serial program, repeating the steps

```
        SAY NEXT NUMBER; POINT AT NEXT OBJECT;
```

within some kind of loop, then it would be impossible to get out of step. But they do, and sometimes spontaneously correct themselves. Adults performing some tasks requiring two sequences of actions to be synchronised, for instance playing a musical instrument with two hands, may experience similar problems.

The ability to run a program in parallel with others, using a third process to achieve synchronisation could be a powerful source of new skills. For instance, it would not be necessary to write a new program interleaving the steps of two old ones, as is required in conventional programming languages. Provided both programs are initially represented in a form which permits synchronisation with messages from other processes, it becomes possible to synthesise a new skill simply by running the two old programs in step. It may be necessary to develop new perceptual skills to check that all is going well, but that would be required in any case for developing and monitoring a single serial program integrating the two skills.

Similarly, instead of re-writing a program to cope with different stopping conditions, the same program could be executed and interrupted by different external monitors: for instance counting all the buttons, counting out buttons till there's one for each button-hole, counting out five buttons. Further, instead of building in error checks, which would have to be different for different uses of a skill (running out of buttons is only an error if you are trying to count out N buttons, and there aren't enough), different monitors for different error conditions could be used for different tasks, while essentially the same basic programs are employed.

If programs are to be run in parallel this can be done either by time-sharing a single processor, or by using a network of processors which can work in parallel. In principle the two are equivalent, though time sharing one processor raises many difficulties if each of the separate processes has its own requirements concerning speed of execution, synchronisation etc. Further, there is plenty of evidence that human and animal brains consist of many units which can do things in parallel. It is therefore most likely that if processes do run in parallel as suggested above, then they probably run on different processors, and are not simply time-shared.

This immediately suggests the possibility that different processors may have different computational resources. For instance they may vary in speed, or memory capacity. More importantly, they may vary in the extent to which they have the capability to run programs or the extent to which they have access to mechanisms required for synthesising procedures, monitoring them, debugging them, interrupting and restarting them, relating execution steps to goals and percepts, and so on.

Thus there might be some processors with all the facilities required for developing and testing programs, and other processors capable only of running the programs. As suggested above, the former processors might make use of comments concerning the purposes of different program steps, which are ignored by the latter (Sussman 1975). If after being fully developed and tested, programs produced by the former are shipped out to the latter processors for execution, then this could produce the kinds of phenomena mentioned above which suggest to many people that compilation has occurred. Our model, however, does not require a major change of representation, such as occurs during compilation, merely a change to a different interpreter.

The more intelligent processor might develop the general structure of a skill or ability, perhaps leaving some of the fine tuning, adjustment of parameters and thresholds, etc., to be done at lower levels while the program is run by a different machine. The latter process would be what happens when an already learnt skill is improved with practice. (I don't pretend to be saying anything about how the fine-tuning, etc. is achieved.)

A theory along these lines could explain how many skills (e.g. musical performance) might be learnt by first learning various subskills which are subsequently put together. The synchronisation of two old skills might involve the development of a new third skill, which will run in parallel with them. (Try opening and shutting your mouth and your fist repeatedly in time. Then try doing it out of phase.) More complex skills might involve an extended hierarchy of sub-processes some of which control others. Some sort of synchronisation between largely independent processes is in any case required for co-ordinating visual perception with movement of limbs.

There are different ways in which synchronisation might be achieved. The difficulty of playing a piano piece where the left and right hand use different beats, suggests that sometimes the co-ordination of two or more low-level machines requires synchronisation signals linked to suitable points in the programs. Synchronisation could make use of global timing signals, shared between all processes. Alternatively, different groups of processes might use their own synchronisation signals. (The former would limit the number of different tasks requiring different rhythmic patterns which could be performed in parallel.) Further, some kinds of synchronisation might use a sort of variable representation of speed (like a throttle), as is suggested by the co-ordination of complex dance movements or the hand and foot movements needed to drive a car.

It is possible that other things besides timing can be co-ordinated. For instance in playing music with two hands, phrasing, stress and volume can be co-ordinated, and the same piece may be played with different superimposed ’expression’, suggesting that there is a supervisory program which controls the way the sub-programs are executed. So besides timing, it seems that at least amplitudes and smoothness of execution can be externally controlled.

If complex actions involve many different processes running in parallel, then interrupting and re-organising the processes may be a very complex matter. Such disturbances seem to play a role in some emotional states, for instance when you lose your balance, or are startled by a face seen suddenly at a window (Sloman 1981).

There are at least two different ways in which a program might be "shipped out" to a lower level process after being synthesised by a "central" processor. The whole program might be copied into the memory of the new processor. Alternatively, there might be common access to some memory, with new processors being told to get their instructions from the very same data-structure built by the program-synthesiser. The former might be more suitable where there is no shortage of processors or memory space, or where there is a shortage of rapid access communication paths. The latter might be appropriate where processors have to be re-used for different purposes, and where subsequent modifications to the program, achieved by the higher-level machine, should be immediately available to the lower levels.

There are many problems and gaps in this theory sketch, including unknown trade-offs. Is there only one program-synthesising machine, or are there several, allowing more than one new skill to be learnt at a time? (E.g. learning a new poem at the same time as learning a new scale on the piano? Learning the words of a song at the same time as learning the tune?) Is there a very large number of processors available for executing programs in parallel, or only a small number (e.g. seven plus or minus two?) The former would allow arbitrarily complex hierarchically organised skills to be developed, subject possibly only to the constraint that a single global synchronising ’beat’ is to be shared between them all. How deep can the parallel process hierarchies get? To what extent is horizontal communication across the hierarchies possible? What happens if the central processor and a low-level processor both attempt to run the same program? (Breathing seems to be an example where this might occur, since it is controlled intelligently in speaking, singing, etc. in addition to being an 'automatic' process.)

Perhaps the running is always done by a lower-level processor, but sometimes under the control of the more intelligent program synthesiser? How are the primitive instructions routed from processors to still lower level processors, e.g. to muscles? If programs are physically copied into the lower level processors, then can processors be re-used during the process of development and debugging a skill? Is there some sort of garbage collection of processors? Similar questions arise about the space required for the alternative system where different processors access the same program stored in the same location. Can storage space for instructions be re-used? How are new processors and new storage space allocated? Do the different processors share limited resources of some kind, e.g. memory or 'fuel', or are they truly independent? Does this hierarchical parallel organisation of "motor" skills also play a role in other abilities, e.g. perception, language understanding, problem-solving?

It is consistent with the model sketched here that many of the lower levels in the human brain use computational resources of types which first evolved in much less intelligent organisms? How did the newer, more sophisticated mechanisms evolve?

What are the implications of all this for our understanding of consciousness? Perhaps if there is a hierarchy of machines, what we are conscious of is restricted to information stores accessible by the highest level system(s). Information and processes occurring in the lower level machines will not be the only things which are not accessible to conscious processing. Information which is in principle accessible will not always be accessed when needed, for a whole variety of reasons, including inadequate indexing, deliberate suppression, etc. It may also be the case that what is accessible and accessed for the purposes of certain sorts of processing may not be recognisable and describable in explicit reports because the system lacks appropriate concepts and descriptive resources, just as we may sometimes not be able to discriminate and describe what is before our eyes until after we have trained in new sorts of expertise.

This is a short list of questions I cannot now answer. I don't claim to have offered a theory. At best it is a research program which may produce explanatory theories one day.

References

J.A. Fodor, The Language of Thought, Harvester Press, 1975.

A. Sloman, The Computer Revolution in Philosophy: Philosophy Science and Models of Mind, Harvester Press and Humanities Press, 1978.
http://www.cs.bham.ac.uk/research/projects/cogaff/crp/
http://www.cs.bham.ac.uk/research/projects/cogaff/crp/crp.pdf

A. Sloman and M. Croucher, 'Why robots will have emotions', IJCAI 1981.
http://www.cs.bham.ac.uk/research/cogaff/81-95.html#36

A. Sloman and M. Croucher, 'You don't need a soft skin to have a warm heart: towards a computational analysis of motives and emotions'.
Cognitive Studies Research Papers No CSRP 004, Sussex University, 1981.
http://www.cs.bham.ac.uk/research/projects/cogaff/81-95.html#55

G.J. Sussman, *A Computational Model of Skill Acquisition*, American Elsevier, 1981.
http://dspace.mit.edu/handle/1721.1/6894