Computer Scientists as Chief Computational Predictors

A Possible Future and How Not to Get There

André Platzer*

1 How Not to Get from CSD50 in 2015 to CSD100 in 2065

On the occasion of CMU's CSD50 semicentennial celebration in 2015, this short note speculates on the role that computer science might play at CSD100 in 2065. A comforting thought that the reader can fortunately hold on to until we get there is that most[1] long-term predictions turn out to be entirely based on misconceptions that will be said 20 years from now should have been entirely obvious to everybody today.

Will CSD100 have to do without computers? At the CSD50 celebration of CMU in 2015, computers already made it in invisible ways into many parts of our lives (phones, coffee machines, washing machines, electronic traction control of cars). It has been a while since I saw a classical desktop computer. Prolonging this trend leads to a CSD100 where computers will be both everywhere and yet nowhere to be found, since computational elements may have become so cheap to manufacture and maintain at low energy cost that they might as well be used in virtually all artifacts. Computer elements are already embedded into clothes. And RFID chips are glued onto arbitrary products. That does not mean CSD100 will moan the loss of computers, but, rather cheer at the (invisible) sight of computational elements that seamlessly float around everywhere in the real world and can be used or combined at will. Just like devices can draw on power from the environment wherever we go today, the devices of 2065 may be able to draw on spare computing power from the environment wherever we go at CSD100. Let's hope that the search for computational adapters that would make that happen will not get even worse than the search for compatible power adapters today.

At the same rate that computers become more invisible, their impact increases. An increasingly large part of a car nowadays is actually a computer on wheels in the same sense that modern aircraft are computers with wings. Some aircraft already fall out of the sky if it wouldn't be for the beneficially stabilizing effects of their computer control. And robots can certainly be considered as computers with arms, or wheels, or legs or other ways of interfacing with their physical environment. That is why these and other systems are collectively known as *cyber-physical systems*, i.e. those that combine cyber elements such as communication, computation, and control with physical elements such as motion.

At CSD100, everything computes. If it becomes easy to integrate computational elements into everything, there is no more need for distinction between computer and non-computer elements since *everything computes* (*panta logizetai* in analogy to Heraclit's *panta rhei*). Everything can read data about its environment, everything can process, everything can optimize, everything can act according to the computed optimum.

At the same time, *something* needs to make sure that the ultimate decision reached that way is in the best interest of the people on account of whom the optimized decision-making is taking place. If computation for such decision-making is happening at the predicted scale, *nobody* will be able to keep up and check whether all those decisions were actually in his best interest. For the large part, one might argue that such checks aren't even necessary since most of those decisions individually won't matter too much. But what about the ones that do? And what about the sum of all individually irrelevant, optimally computed decisions on our

*Computer Science Department, Carnegie Mellon University, Pittsburgh, USA, aplatzer@cs.cmu.edu

[1] Even more fortunately, this short notes does not leave enough space for references so that the thoughts in this thought-piece do not have to be slowed down by the burden of distinguishing citable fact from mere conjecture.

behalf? Or what about decisions by algorithms trying to prevent a sleepy driver from causing an accident with his car? What about a program helping a pilot avoid a collision with an almost invisible unmanned aerial vehicle close by?

From Computer Science to Program Science? If everything computes since computational elements are everywhere, then their most important ingredient will be the program. Today, it is already the software that has a major and growing share in the added value for cars, yet also in the development and debugging cost [2]. If literally everything computes, there may not be enough people to write the corresponding programs with the detailed attention needed to make sure the program does exactly what it is supposed to. The thought that the programs ruling tomorrow's real physical environment may be as badly written as some of today's cell phones or web pages might make one shiver. Will this lead to more automation surprises, well-documented, e.g., in aerospace [9], except amplified to all aspects of our lives since everything computes?

From Program Science to Intention Science? If there are more programs (or spontaneous combinations of programs into conglomerates that compute together to meet various objectives on behalf of their "users") than people to program or monitor them, then it will become even more crucial than today to make sure the program meets the programmer's intent. Did the program compute what we wanted it to compute? Or did it come up with an answer that—while equally plausible to the (often absent) observer—is plain incorrect? How can we tell? How do we even specify what exactly we wanted it to compute in the first place?

Ubiquitous programs could also mean that many programs derive from higher-order directives with more emphasis on what needs to be done and less emphasis on the means to get there. Whether those intent declarations will be by precise specification or by example depends on the application. With ever more programs for ever more complex tasks, it can be expected that there will be more programs that base their decisions to a larger extent on the result of their own learning than on the program itself. How do we say what our intentions for learning programs are? How do we check the outcomes? Engineers now compute and selectively discard results, e.g., of numerical solvers for mechanics [1, 3], if the results are incompatible with physical expectations. They then rerun with different randomizations to get other answers. As computations get more complicated and as grounding in physical intuition may become ambivalent, are we doomed to believing the outcomes of incorrect computations? Or is there a principled way of checking the answers?

If the most important ingredient of a computational element is its program and the most important ingredient of a program is its intent, will the primary responsibility of computer scientists shift to defining intents for programs? Or to checking whether those intents are met? Will that responsibility be shifted to computational lawyers that retroactively investigate which program misbehaved compared to its intent based on the forensics of the respective log files? Or is there a more proactive way of making sure that programs can't go wrong in the future, at least the ones that reach critical decisions such as driving cars for us? At least some results indicate that there is hope for a disciplined answer about the correct outcomes of systems [7], even complicated ones like the next-generation airborne collision avoidance systems ACAS X with its heavy use of nontrivial optimization [6].

From Computer Science to Computed Science? While the term "*computed science*" will hopefully not stand the test of time, it directly emphasizes an increasingly important aspect of computer science. Of course, computer science is not and has never been the science of computers. Yet, "computed science" would emphasize the evolution toward the science of assessing what has been and what will be computed. The key question is: *is the actual value that has been computed the correct value to base a decision on?*

Like the call to action that is (more reasonably) expressed as *computational thinking* [10], the term "computed science" observes that computations play an increasing role in the other sciences, where programs begin to be perceived as as instrumental as they really are for the progress of science [4, 5] and engineering [1]. At the same time, a number of scientific would-be results stumbled over mistakes in their software [5].

Computer Scientists as Chief Computational Predictors. Where would this development lead the computer scientists of 2065? And what could they have to offer to solve the challenges? It seems like predicting or assessing the quality of a computation may become the primary responsibility of computer scientists. Besides predicting the runtime duration and required resources of a computation, it is crucial to predict or check correctness of its results. The task itself will not be any easier than it is today since scale creates more aspects of unreliability: unreliable components, unreliable communication, failing computational elements. But it will become even more crucial with computation that becomes more cyber-physical [8] as computer and physical elements blend seamlessly.

Computer scientists might even become like the N in "P vs. NP": They may become chief computational selectors using their computational complexity experience to influence algorithms to select the right solution path. Dually, computer scientists are chief computational validators and in charge of checking or ensuring the success of a computation. It's about time that computer science develops stronger analytic tools to help them assess the quality of a computation to enable the revolution that "computation everywhere" foreshadows with the responsibility that its real-world impact requires.

References

[1] Michael Beer and Martin Liebscher. Designing robust structures - a nonlinear simulation based approach. *Comput. Struct.*, 86(10):1102–1122, May 2008.

[2] Manfred Broy, Ingolf H. Krüger, Alexander Pretschner, and Christian Salzmann. Engineering automotive software. *Proc. IEEE*, 95(2):356–373, Feb 2007.

[3] Frank C. Günther, Heiner Müllerschön, and Willem Roux. Robustness study of an LS-DYNA occupant simulation model at DaimlerChrysler commercial vehicles using LS-OPT. In *8th International LS-DYNA Users Conference*, 2004.

[4] Erika Check Hayden. Rule rewrite aims to clean up scientific software, 2015.

[5] Darrel C. Ince, Leslie Hatton, and John Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485–488, 02 2012.

[6] Jean-Baptiste Jeannin, Khalil Ghorbal, Yanni Kouskoulas, Ryan Gardner, Aurora Schmidt, and Erik Zawadzki André Platzer. A formally verified hybrid system for the next-generation airborne collision avoidance system. In Christel Baier and Cesare Tinelli, editors, *TACAS*, volume 9035 of *LNCS*, pages 21–36. Springer, 2015.

[7] Stefan Mitsch and André Platzer. ModelPlex: Verified runtime validation of verified cyber-physical system models. In Borzoo Bonakdarpour and Scott A. Smolka, editors, *RV*, volume 8734 of *LNCS*, pages 199–214. Springer, 2014.

[8] André Platzer. Logics of dynamical systems. In *LICS*, pages 13–24. IEEE, 2012.

[9] Nadine B Sarter, David D Woods, and Charles E Billings. Automation surprises. In *Handbook of Human Factors and Ergonomics*, volume 2, pages 1926–1943. 1997.

[10] Jeannette M. Wing. Computational thinking. *Commun. ACM*, 49(3):33–35, 2006.