Rant on Turing

John Doyle

Motivation

Nature just had a special issue with essays on Turing and most of them were awful (but short, so I recommend reading them anyway). Sydney Brenner's was good though. Anyway, what can be done to fix this and write a better paper or collection of papers using Turing as a starting point? The immediate target audience would be roughly typical Nature readers, but this would hopefully be relevant to teaching systems theory in engineering (and physics and biology) in a more integrated way to a younger (e.g. early undergrad) audience. This is somewhat motivated by Turing's 100th birthday, but I think it might be natural anyway.

So here is a high level summary of what I have in mind, with some explanatory background that can hopefully be pointed to elsewhere…

What I basically want to explain (and need serious help on as I have major holes in my understanding) is

- what Turing was really about beginning with the 1936 paper but continuing thru all his work,
- what engineers and mathematicians take too for granted about Turing's ideas, and don't explain
- and scientists completely miss (see *Nature*),
- with particular focus on the relevance these ideas have for a more integrated theory of "complex systems" and systems biology and neuroscience

Other directions not adequately discussed here but which are priorities (and will be discussed in the first 213 class). To repeat a familiar platitude, we need a fundamental redo of engineering systems theory (comp, comms, control), but this, time, seriously do it. Starting from Turing we can go in a few orthogonal directions that start showing what the space looks like:

1. redo Bode and Shannon to look more like Turing (this is pretty standard)
2. redo Shannon as stochastic relaxation of intractable comms (Kolmogorov) problems (also standard)
3. put together clearly all the mixes of Bode, Shannon, and Turing that have been done so far, highlighting current holes that might be closeable
4. redo stat mech using this style (mostly done but needs exposition, QM needs lots of work)

For purely pedagogical reasons, using Turing more directly might be useful:

5. see how much of control theory can be done with automata and TMs and then transition to Bode
6. see how much of statistical physics can be done with automata and TMs and motivated by analog to digital, and then down via active to passive to lossless to distributed to quantum.
7. Then see if we can tie these together, perhaps only in their automata versions? The automata angle is only for pedagogical value, so not the highest priority.

- A current challenge is that each subdiscipline (computing, controls, comms, physics etc) tends to make assumptions about their models and architectures that are incompatible, and progress on "hybrid" theories have been slow, though as mentioned in 3 there are some results. Hopefully this is not something impossible to overcome, and I'll try to address this in another rant and in class.

The longer term goal for which this would be a first baby step:

Starting with Turing, we would do a thorough rethinking, refactoring, and ultimate integration for a new theoretical foundation of complex systems that are now fragmented and stovepiped in computing, communications, controls, OR, and statistical physics (and various application domains such as networking, biology, neuroscience, medicine, economics, etc). The idea would be to go back to Turing and rethink it all in light of what we now know. Hopefully the reason for starting with Turing will become clearer, beyond this being his 100th birthday.

I was originally hoping that we could take the stovepipes and somehow integrate them directly, but unfortunately that seems impossible. The stovepipes are too rigid and assume incompatible architectures. The good news is that there seems to be a finer level of granularity that could be used to build a new more coherent foundation, but this will take much more than a simple union, but a lot of basic research on what amounts to almost a whole new subject. Experts in existing stovepipes might initially resist even temporary reconfiguring of their subjects, but my experience so far has been encouraging, as the need for greater integration and the failures of direct approaches is now obvious to almost everyone.

A bigger problem is that much of what has been done in "new sciences" of complexity and networks has been worse than nothing in muddying the waters. Hopefully we can learn from the mistakes made and avoid them in the next attempt. Recent progress here too has been very encouraging. But now for a step back to the 1930s…

Background to put what follows in context

The standard account of Turing's contributions, which I hope I can mostly take for granted, a subset of which is:

1. Theory (1936): Turing machine (TM), computability, (un)decidability, universal machine (UTM)
2. Practical design (early 1940s): code-breaking, including the design of code-breaking machines
3. Practical design (late 1940s): general purpose digital computers and software, layered architecture
4. Theory (1950): Turing test for machine intelligence
5. Theory (1952): Reaction diffusion model of morphogenesis, plus practical use of digital computers to simulate biochemical reactions

(Ignoring here the precursors of neural nets, presumably they will be highlighted by someone).

In all cases he was ahead of his time, and produced results of both immediate and lasting value, deep but with broad impact. In breadth, depth, impact, significance, lasting value, this (arguably) is collectively the most significant collection of achievements of the 20th century, maybe since Newton, maybe ever. (He was also a world-class marathon runner, and a man of the highest integrity. Ironically, he was not just the smartest, but in some sense one of our age's most heroic and moral.)

In our fragmented and siloed world, these achievements are poorly understood, but the standard accounts certainly capture his astonishing accomplishments, if in a somewhat fragmented way.

What I claim is missing is an appreciation of the unity and continuity of the work, the deep connections between the theory and the practical, the computational and the biological. Perhaps only in the last 10 years have we finally fleshed out the theory, technology, and biology sufficiently that the rest of us can finally see what Turing was really getting at all along, and restart a more coherent research program along the lines that I think he would suggest now.

Put less euphemistically, we mostly still don't "get" Turing, but now could. For engineers and mathematicians, he is so deeply and pervasively embedded in the complex fabric of math and technology that he is taken for granted like no one else. For most scientists, e.g. readers/authors of Nature, they don't understand either the math core of the theory, or the case studies in (decades of) technology and (very recently) biology that now confirm his insights about complex systems. Plus to the extent they think about complexity at all, they have been fed a view of (the "new sciences" of) complexity that is in direct conflict with Turing's insights and the attention to these unrewarding views of complexity has distracted many fields from appreciating Turing's work.

Turing died just when it is likely he would have started putting a coherent picture together, blending the progress in molecular biology, neuroscience, and particularly technology, and drawing them together, even as they were fragmenting into reductionist stovepipes. He likely would have also integrated the works of Shannon, Bode, Wiener, the AI community, the OR community, etc… which were briefly in contact in the 1950s mostly under the banner of cybernetics but shattered into fragments by 1960 (another tragedy different in detail from Turing's but similar in its unfortunate blending of private and public), and have only begun to come together in the last decade in the context of theories of the Internet and systems biology. Engineers coped with this fragmentation by building highly modular systems that needed only a few systems architects to pull the whole systems together. This approach delivered dramatic performance and features but fundamentally unsustainable infrastructure, while scientists spiraled off into "new sciences" of complexity and networks, which are at best, misguided. For some background on this see:

- Alderson DL, Doyle JC (2010) Contrasting views of complexity and their implications for network-centric infrastructures. IEEE Trans Systems Man Cybernetics—Part A: Syst Humans 40:839-852.

Turing, chaos, and complexity

It is interesting to note that Turing dismissed the centrality of chaos to a theory of complexity (even though it was not called "chaos" in 1950…):

> "… quite small errors in the initial conditions can have an overwhelming effect at a later time. The displacement of a single electron by a billionth of a centimetre at one moment might make the difference between a man being killed by an avalanche a year later, or escaping. It is an essential property of the mechanical systems which we have called 'discrete state machines' that this phenomenon does not occur. Even when we consider the actual physical machines instead of the idealised machines, reasonably accurate knowledge of the state at one moment yields reasonably accurate knowledge any number of steps later."
>
> Turing, 1950, Computing Machinery and Intelligence, *Mind*

Ironically, since Turing, within science "nonlinear dynamics" has become essentially synonymous with "chaos" and thus 'unpredictability', whereas to Turing and all engineers since, nonlinear dynamics are absolutely essential elements in building the (almost) perfectly predictable and repeatable "discrete state machines" (e.g.

digital hardware) from the potentially chaotic analog electronic substrate. Indeed, it is the creation of sufficiently nonlinear active devices that is always at the heart of digital design, no matter what the physical substrate. The vacuum tube and transistor are only the most familiar examples. This issue will pop up again when we reconsider the 1952 paper.

Importantly, this quote dismissing chaos as irrelevant connects to both the 1936 and 1952 papers:

- Turing machines and their implementation depends on "this phenomenon" (of chaos) not occurring and perfect (if necessary after suitable error correction) repeatability of the machine dynamics
- Turing patterns (1952) can be viewed as a very first cut in how to achieve this using the physical substrate of biochemistry instead of electronics. Turing explicitly said that it was the discrete machine not the electronics that were essential, particularly in his 1947 lecture to the LMS, quoted here:

1947 Lecture to LMS:

> "… being digital should be of greater interest than that of being electronic. That it is electronic is certainly important because these machines owe their high speed to this… But this is virtually all that there is to be said on that subject. That the machine is digital however has more subtle significance. … One can therefore work to any desired degree of accuracy. This accuracy is not obtained by more careful machining of parts, control of temperature variations, and such means, but by a slight increase in the amount of equipment in the machine."

These two quotes are significant:

- The complexity ("amount of equipment") in the implementation of digital hardware is to create their near perfect repeatability that we are all familiar with. This is a special instance of the general theme that much of the complexity in implementations is to create robustness, not minimal functionality. Here minimal functionality is being digital, and robustness is the repeatability and accuracy of this digital "abstraction" (a word we will return to) of an underlying analog substrate, despite (possibly substantial) uncertainty in that substrate.
- We don't have a clean theory of exactly how much increased complexity and energy consumption is required to make a robust digital virtual machine out of real analog circuitry. There is some work, and it should connect to error correcting codes. Need help here.
- Nonlinear dynamics play an essential role in all implementations of digital logic, electronics being the most familiar, and this can be proven to be necessary (this could be made clearer in our work), but chaos per se plays no role whatsoever
- Interestingly, analog circuits implementing digital logic might appear chaotic to a naïve observer unfamiliar with their function. And random analog circuits statistically similar to those implementing digital logic probably would be chaotic, and this could be interesting pedagogically.
- The related idea that functional networks are at the "edge of chaos" is wrong, but nevertheless interesting, given its popularity and role as the first "big idea" in "complexity science"… It would be nice to put this all in perspective but it is only for pedagogical purposes.
- There has been some work connecting chaos and undecidability but I don't know it… something to look into perhaps… New book by Chaitin might be relevant.

Some essential observations that need explanation and compression en route to the main punchline

There is a style to Turing's work that has become the paradigm of modern complex systems research in engineering and recently in systems biology (but emphatically not yet in science), which doesn't yet have a common set of terminologies, but involves 3-4 "universal" elements:

0. **Virtual machines:** Define an abstract model, here the Turing machine, given plausible constraints on available or envisioned components and descriptions of system requirements. The machine is idealized so any limits on its capabilities are likely to hold for less idealized machines, but this is a subtle issue.
1. **Universal laws:** Hard limits (constraints, laws, tradeoffs) on what is ideally achievable for the virtual machines
2. **Universal architectures:** Abstract architectures (i.e. protocols, layers, virtualizations, universal machines) that achieve these hard limits and allow users of the architecture to find problem-specific, suitable tradeoffs, often in highly modular ways
3. **Implementation**: Practical, implementable designs that demonstrate the relevance of both the tradeoffs and the architectures to real systems

Turing's early work on computers typifies this (assuming for now that the reader understands or is at least somewhat familiar with this, so review it but connect it a bit with modern terminology):

0. The key virtual system is the Turing machine (TM) which is an abstract version of an arbitrary algorithm running on arbitrary digital hardware ("discrete state machines") with an infinite data memory. Turing for the first time clarified what an algorithm was (Gödel's commentary on this is important). Don't need to be strictly historical here as modern accounts are more accessible than Turing's (or Gödel's).
1. The theory of (un)decidability sharply distinguishes what can (and cannot) be computed by even idealized algorithms and even in principle. Turing shows that the (un)decidable boundary does not depend on a large set of variants of the basic TM, suggesting it is a quite fundamental (i.e. universal) law (constraint, hard limit) on computation. (Mathematically, this restated Gödel's incompleteness theory in a more transparent way and showed its relevance to computation. It also connected back to Cantor via computable numbers. Strangely, Turing, Cantor, and Gödel all appear to have committed suicide.)
2. The "universal Turing machine" (UTM) and related theory proved that the modern computer architecture consisting of a suitable combination of digital hardware (a TM) and software, can actually implement *any* algorithm from 1). Given a TM and data, both would be put in memory, and the general UTM would then emulate the special TM. In other words, a system with this UTM software/hardware architecture can compute all that can (even in principle) be computed on any TM. This is a new virtual machine with TMs in software in a UTM.
3. Practical designs using existing (analog) electronic components of a general purpose digital computer implementing the universal Turing (now misnamed Von Neumann) architecture, as well as early methods to develop software

By making an algorithm (the TM) part of the data, Turing "invented" software, which opens up the opportunity for further virtualization and layering. (He also was working on practical aspects of software.) That is, algorithms can treat other algorithms as data, so operating systems, compilers, interpreters, higher level languages, layered networking protocols, etc all become possible by recursively applying the basic initial layering (or virtualization) idea of the UTM. Within computing and networking, this role of layering and virtualization has most dramatically proven to be the essence of architecture, but other disciplines, notably biology are just starting to see this as well. An attempt to explain "layered architectures" to nonengineering, and particularly neuroscience, audiences is:

- Doyle, ME Csete (2011) Architecture, Constraints, and Behavior, *P Natl Acad Sci USA*

So a major element of a rethinking of controls and comms theory is to be more explicit about the 3 steps, and also to start combining them more into one theory, still iterating on the 3 steps. This obviously needs a lot more explanation.

Universality = deconstraint

There are also several senses in the word "universal" here. One is that this 3 step strategy seems like a universal approach to systems theory, another is that the first step involves constraints that could be called "universal laws," and the second step involves "universal machines" which is Turing's main use of *universal*. The third step could also involve *universal* as well in that some specific architecture (e.g. the bacterial biosphere, see below) could become universally shared by all cells.

The notion of *constraints* is a convenient way to organize this discussion and make the verbal descriptions more naturally map onto the mathematics (see Doyle, Csete 2011). The universal laws are hard constraints that are intrinsic to the problem, the generalization of natural laws such as energy conservation to complex systems. The universal architectures are also constraints but are in some sense chosen to achieve this universality. It is necessary for a machine to satisfy these highly restricting constraints only to achieve universality, which is a kind of robustness to arbitrary problems. Successful (e.g. universal) architectural constraints are "constraints that deconstrain" (Gerhart and Kirschner) in that adopting the constraints then *deconstrains* what problems can be solved with systems using the architecture. Universality is the ultimate in deconstraint.

Terminology: abstract versus layered, virtual, and universal

Looking back from our now heavily digital world, the role of the digital "abstraction" is crucial and could be the starting point for a more coherent theory of everything from layering to multiscale physics. But we might consider replacing the overarching term "abstract" with "virtual" or "virtual machine" (a la Sloman), and more generally with layer, virtual, and universal. While *virtual* has many diverse meanings from the narrow use of virtual machine in operating systems, to the more slippery use of "virtual reality", the term *abstraction* is even more heavily overloaded.

Using virtual (+layer+universal) in the context we have here might help some, but Turing didn't use this term (he only used universal) so it needs some thought. Systems can be layered with little virtualization and no universality. Concretely, the Internet has a layered architecture, but IP addressing is not virtual (a huge flaw given its modern usage). Many user interfaces involve elements of virtual reality in the sense that they create the illusion of the "real world" but are rarely universal in the sense of being able to create any environment experienced in the real world. So layering and virtual and universal all mean slightly different things, and perhaps the only necessary implication between them is that universal implies virtual. And I'm not even sure of that.

So I'm suggesting (tentatively) using the term *digital virtual machine* in place of what we now might call the *digital abstraction* of the underlying analog circuitry. (Though usually I'll just refer to these as digital or analog.) The notion of virtual machine then recurses naturally (as does abstraction) downward into active virtualization of passive circuits, and upward into software/hardware and applications/OS. We naturally speak of software layer on hardware layer, and application versus OS layer, but tend not to say digital layer on/in analog. The latter seems natural, but needs more careful explanation. Indeed, the exact nature of digital/analog and software/hardware is extremely well understood by experts, but this is never explained to others, and the virtualization itself makes the hidden workings deliberately cryptic.

I also want to distinguish between *virtual* and *universal*, as it is possible to be the former (e.g. a TM is a virtual machine when thought of as a digital virtualization of analog circuitry) without the latter (a UTM is a very special kind of TM). It appears that only virtual machines can be universal but perhaps there are contexts in which "universal" in this sense does not depend on virtualization. In computing and biology, virtualization seems to be essential to universality. I'm not aware of anyone who has addressed this issue.

So we might rephrase Turing (this needs work) by saying that a TM abstracts what it means to decide and/or compute, and that digital logic is a virtualization of analogy circuitry that is universal in the sense of able to implement TMs. The UTM is both a *virtualization* in software of the TM hardware, and *universal* in that any TM can be implemented as software on a UTM. (It is thus possible to imagine a VTM that is not a UTM, by considering a VTM that virtualizes some subset of TMs but not all TMs.)

Layered architectures

The implications of layered (e.g. Turing) architectures are so familiar to everyone that they are taken for granted, though in different ways by engineers versus users, though part of the architecture is that everyone is a user, since designers use computing infrastructure to design new infrastructure. Though engineers worship Turing, they barely know what he actually did, since its success is marked by its pervasiveness. So much has happened since Turing died that we forget how much is due to him.

But for the user of modern information technology (IT), they can download and immediately run a bewildering variety of application software or "apps" with the only compatibility requirement being on the operating system (OS) on their smartphone or PC. Different OSes and their hardware platforms are both now quite flexible in their internal implementation, but there are only a handful of available alternatives. These OSes can also connect a vast diversity of hardware, again only limited by OS compatibility, an example of an "hourglass" layered architecture. But all the details are hidden or abstracted (this usage seems too ambiguous) or *virtualized* (this is the usage I am advocating), from the user. The same holds for the layers of the Internet architecture.

For programmers or "app" developers, there are rich sets of tools (e.g. compilers, higher level languages, code libraries, application interfaces, etc) that allow them to exploit the layered architectures of modern IT systems. Unfortunately, the architecture that promotes all this "plug and play" modularity also is vulnerable to viruses and worms in a way that unlayered digital hardware (without software) is not. Many recent dramatic examples.

Most scientists have used higher level languages or at least domain specific scientific computing software, but need to remind them that the ease of all this is only because of the Turing architecture that layers software of applications on operating systems, then software on digital hardware, and digital on analog, etc. as well as layered network protocols (the Internet protocol stack).

The divide in understanding between users, even the PhD scientists who are "first adopters" of modern technology and the engineers who design it, is unbelievably huge. (The modern education of engineers is a disaster, and creates a collective kind of inarticulateness that makes them apparently incapable of explaining how anything really works. But that too is overly polemical, and at least some of it is probably my fault, and I should shut up and fix it rather than whine. Hence attempts like this need attention.)

For example, the Internet "architecture" appears to be a source of complete confusion and bewilderment to scientists, and *Nature* (and *Science*, to be fair) has arguably contributed mightily to this. Probably best to avoid this issue for now, but it eventually needs to be addressed. Alderson and Doyle (2010) has an overview of the ongoing persistent errors and confusion (*Nature Physics* just had a special issue on complexity that is a case study in error and confusion), plus references to the details (appearing in *PNAS* and *Notices of the AMS*, e.g.)

- Doyle et al, (2005), The "Robust Yet Fragile" Nature of the Internet, *PNAS* 102 (41), October 11, 2005
- Willinger W, Alderson D, and Doyle JC (2009) Mathematics and the internet: A source of enormous confusion and great potential. *Notices Amer Math Soc* 56:586-599.

Persistent errors and confusion

There seems to be two opposite misunderstandings of architectures that are layered and/or virtualized. One is exemplified by creationism, where the gap between nonlife and life, or between animals and humans, (why not transistor and the Internet) seems so vast a gap that only supernatural forces can explain them. (This is called the "god(s) of the gaps" and produces a highly time-varying notion of deities as the gaps shift.)

The other extreme is statistical physics and its application to complex systems (SPCS), where the gaps are assumed to be so small that they can be covered by largely random ensembles and minimal tuning, as exemplified by such concepts and theories as "order for free, edge of chaos, self-organized criticality (SOC), scale-free networks" etc etc. This is most perfectly illustrated by the use of SOC as a model for the brain, with the claim being that understanding simple cellular automata models of sandpiles is tantamount to understanding the human brain (see Chialvo in Nature Physics). This almost makes the creationists seem sensible.

I think this "small gap" SPCS prejudice helps drive connectomics and certainly the NetSci approach of Bullmore and Sporns. That understanding the connectivity in any layer will help understand the whole... more on that later...

The gaps between digital/analog, software/hardware, etc are huge but have fairly clear (if wildly misunderstood) explanations, so potentially can serve as a starting point for helping scientists understand multiscale layered systems in biology and technology. Creationists seem not to doubt what engineers tell them about their PCs and cell phones (but they often doubt mainstream accounts of global warming), whereas SPCS has been applied everywhere with almost comical results (until you realize the publications have such high impacts, and are now being pursued in medicine and neuroscience with potentially negative consequences).

We have had little success persuading either the creationists or SPCS faithful that the gaps are explainable with appropriately conceptualized virtual machines (though the origins of life remain do largely a mystery since the evidence was long ago wiped clean, but even here progress is encouraging). This is unlikely to change the minds of the faithful but we can hopefully help the agnostics to better decide what faith to adopt or (better yet) avoid. It is also should help to offer a coherent alternative rather than just saying "it's more complicated than that."

While the full story is complicated and a real case study in the psychology and sociology of science, the main source of confusion recently in science regarding architecture is that it is synonymous with graph topology, and modularity with weakly linked subgraphs that are internally tightly linked. (This is the most recent incarnation of the SPCS approach.) In fact, architecture and modularity are most importantly the opposite of this. (Just as it is true that even mildly nonlinear dynamics can be chaotic and thus unpredictable, the most perfectly repeatable systems are necessarily extremely nonlinear, as in the circuitry that implements digital logic in CMOS VLSI.)

Ok, back to what topology (whether connectomes at the neural level or Bullmore/Sporns at the higher levels) is very unlikely to do...

As a concrete example, an application running on the Internet (e.g. a social network, a hyperlinked document browser like the web, etc) is allowed to have an arbitrary topology, and indeed this freedom (this "deconstraint") is exactly the point of a layered architecture. Anything goes. Similarly, the physical hardware resources, hidden by OSes and their networking extensions that virtualize the raw hardware resources, have the freedom (deconstraint) to have quite arbitrary topologies provided that they suitably implement the required protocols. Thus topologies, whether in software or hardware, are by design the least constrained aspects of a system implemented with a layered architecture. So the thing that appears easiest to measure, low level neural connectivity, or high level fMRI correlations, are almost certainly the least informative about what is most essential to how brain makes mind, which are the hidden intermediate layers (analogous to the OS instead of the transistors or the app software).

Instead it is the layering itself and the protocols that are universal and thus constrained that are the essence of architecture. This has proven almost impossible to explain to scientists generally, though some progress is being made. Engineers take this so for granted they don't create accessible tutorial material.

The most important aspect of "modularity" in a Turing-inspired architecture is also the role of layering (and virtualization). Concretely, the most important modularity feature is the split between software and hardware, but this split is not at all one of "strong internal, weak external" links between subgraphs. Again, it is quite the opposite. Software is so deeply intertwined with hardware that it might not be obvious to an observer without a priori deep understanding of the architecture, just what the software was. That is, in a UTM, without expert knowledge, you wouldn't know it wasn't just any old TM, and would miss the most important idea in "modularity" perhaps ever thought up. The encoding of algorithm plus data just looks like data. No weakly connected subgraphs. This issue is so embedded in our understanding of software/hardware as well as digital/analog, that we seem never to explain it to the scientists.

So both "architecture" and "modularity" in the modern sense were essentially invented by Turing (though building directly on Godel's theory and lots of engineering for decades or centuries), and have nothing to do with what these words mean to scientists.

This crops up in discussions of the brains modularity. At the sensorimotor periphery there are obvious mappings of fairly nonplastic functionality on the hardware but higher up the functional modularity is less mapped onto obvious anatomy, and this is also true of robots, as you get into higher levels of control and decision making, it's all in software and doesn't map onto specific hardware, whereas sensorimotor peripherals do.

The Turing-style notions of architecture and modularity (and nonlinear dynamics)

Unfortunately, a theory of layered (as optimization) architecture suitable for the Internet architecture has only been developed in the last decade, the above Alderson and Doyle paper refers to it, with lots more details in references. The Internet is a case where the practical architecture of step 3 preceded the theory in steps 1 and 2 by decades. Indeed, Turing is a remarkable instance of doing the theory first, though technology for decades and centuries anticipated his ideas in many ways (that I can't hope to describe). Usually engineers find solutions that the theorists later formalize and extend, returning to engineers tools that enhance future designs. That is, engineering *theory* is often first about "reverse engineering" existing systems and architectures (step 3 is first) to formalize their essential features, leading to steps 1 and 2, but then another round of 3. Engineers always go beyond the theory, so this process recurses.

Turing (and to some extent Shannon) are singular counterexamples to this trend. They did theory without first formalizing existing practice. (Note also that science has always been applications of technology and its mathematics to the natural world, not the other way around. But that's another long story that is full of persistent errors and confusion.)

What is remarkable is how general (in retrospect) the Turing pattern of research and development is: Virtual machines and hard limits on not just computation, but measurement, prediction, communication, decision, and control, as well as the underlying physical energy and material conversion mechanism necessary to implement these abstract process are at the heart of all modern mathematical theories of systems in engineering and science (often associated with names in addition to Turing such as Shannon, Poincare, Gödel, Bode, Wiener, Heisenberg, Carnot,…). They form the foundation for rich and deep subjects that are nevertheless now introduced at the undergraduate level. Unfortunately, these subjects have remained largely fragmented and incompatible, even as the tradeoffs *between* these limits are of growing importance in building integrated and sustainable systems. I think had Turing lived, he would have resisted this fragmentation. We need to "channel" him now.

Recent progress

An essential research direction that has finally materialized only in the last decade is an increasingly integrated theory of hard limits based on optimization that deals systematically with uncertainty, robustness, and risk in complex systems. Basically it is control theorists who have started to incorporate Shannon and Turing more explicitly. This has corresponded to a remarkable convergence between our understanding in biology and technology of what are the potential universal laws, architecture, and organizational principles. As a recent introductory and accessible example, see

- Chandra F, Buzi G, Doyle JC (2011) Glycolytic oscillations and limits on robust efficiency. *Science*, Vol 333, pp 187-192.

This is the tip of the iceberg, as biologists are articulating richly detailed explanations of biological complexity, robustness, and evolvability that point to universal principles and architectures that are surprisingly familiar even if the terminology is different. So basically, the biologists (or biology itself) has already produced spectacular architectures as in step 3, and we are now reverse engineering these, and this *Science* paper is an example of interpreting existing biology (and a central mystery of biology) in exactly the style that Turing used in step 1 (with a bit of step 2).

Hopefully more theorems and case studies like this are to follow. This is in part just due to increasingly detailed description of components but also a growing attention to systems in biology and neuroscience, and so the organizational principles of organisms and evolution are becoming increasingly apparent.

In addition, while the components differ and the system processes are far less integrated, advanced technology's complexity is now approaching biology's so there are striking convergences at the level of organization and architecture, and the role of layering, protocols, and feedback control in structuring complex multiscale modularity. As I said, readers are familiar with the symptoms of this but not the causes. Engineers understand the details but explain the essence badly. We still need Turing.

Determining what is essential about this convergence and what is merely historical accident requires this deeper understanding of architecture — which in general is the most universal, high-level, persistent elements of organization, exactly as Turing described (though he doesn't call it "architecture") — and protocols. Protocols define how diverse modules interact, and architecture defines how sets of protocols are organized. The paradigmatic examples of protocols, layering, and architecture begins with the "universal Turing machine" of separating software and hardware, and describing protocols that connect a specific algorithm and the universal machine.

The obvious symptoms of all this "architecture" and its supporting theory and methods are the extremely evolvable computing and information technology environment we are immersed in, as well as the hidden but no less dramatic impact that advanced control systems have had on automating our cyber and physical infrastructure and machines. This needs explaining to highlight the connections with Turing…

Architecture in biology, HGT, layering, virtualization, universality

What has not been explained adequately, and can only be hinted at is the technical details in biology that mirror those in technology. The simplest to explain now is the microbial biosphere, which in the latest view involves a complex "Internet-like" (Caporale) orgy of horizontal gene transfer (HGT). Organisms swap everything from genes to whole networks, so that the "tree of life" has been replaced by a "web of life". (Shapiro, Koonin)

That is, the Chandra et al Science paper does a step 1 of Turing, finding the hard limits that are really driving the architecture in biology, which in this case is tradeoffs between robustness and efficiency. But step 2 is only minimally addressed. What is needed next is a full Turing step 2 and an explanation of how the whole cell architecture beautifully trades off robustness and efficiency, up against hard limits. I've mostly worked all this out and vetted it with biologists in some detail, but it will take a long major effort to write it up in an accessible way. The details are much more complex, not surprisingly, than anything like it that has been done before.

But Turing would have naturally guessed what is necessary to make this HGT world possible: a "universal machine" or in modern terminology a universal machine using a layered, virtualized architecture. Here is it will be important to carefully explain the relationships between the terms layer, virtual, and universal.

What does that mean exactly in biology? First, we don't know exactly what reactions and networks of reactions can be catalyzed by protein enzymes, so we can't completely finish step 1 yet (I can say more about that but want to go on to step 2), but we can say that prokaryotic cells have a universal and virtual and layered architecture of replication, transcription, translation, and control (e.g. one, two, and multicomponent signal transduction systems) that means that all possible reactions and networks can be implemented with this architecture, provided the suitable genes and promoters can be obtained. Just as the smartphone and PC user can download any app compatible with their OS, where there are only a few. Prokaryotes have only one shared architecture, though the detailed implementation in various polymerases, ribosomes, and other proteins can differ significantly, just as with Turing and modern IT systems.

Note that we haven't even gotten to Turing patterns….which is the final punchline… but apparently nobody has noticed that HGT only works with a shared, layered architecture, which Turing pioneered. It is taken too for granted to be noticed. Let's stop to distinguish what we mean here by these terms:

Layer: The overall reactions in the cell can be broken up into replication, transcription, translation, and metabolism/control. This is adapting the existing terminology in biology to describing a layered architecture, and the fit is remarkably good. These are (importantly) virtual layers, in that they are described in terms of abstract reactions.

Virtual: The reactions that make up the layers are virtual, and are implemented by being catalyzed by enzymes. It seems like this issue has never been properly explained. Need to work on this. Virtual is important because there are (an almost infinity of) multiple ways to implement the reactions that define the layers and hence the architecture.

Universal laws: See Chandra et al. Tradeoffs on robust efficiency. What we need to do is extend the analysis to the whole protocol stack. This is discussed below.

Universal architecture: As a whole the layered architecture is universal in the sense that any set of enzymatically catalyzed reactions can be implemented with the appropriate genes, adequate supply of energy and raw materials, and the right initial conditions. All of these are quite intricate and must be finally organized and tuned.

Universal implementation: There is another sense in which the virtual, layered architecture is "universal" and that is that it is the same for every cell on the planet. This is a problem because these two notions of universal are different and probably should have different names. I confuse them in my writing all the time.

So the first punch line is that the first known "universal virtual Turing-like machine" is the layered architecture of the prokaryote biosphere. I think this is a huge insight and connection that seems to have been completely overlooked. I've been sort of pushing it for a few years but haven't gotten around to explaining all the details. Hopefully I'll get to it soon, but the basic idea isn't that hard to get and can be explained minimally as above.

The much bigger punchline connected to HGT and other mechanisms that cells use to rewrite their genomes on the fly, is that the layered architecture of rep, transc, transl, signal transduction provides control mechanisms in each layer, and this allows exquisite tradeoffs between efficiency (e.g. metabolic overhead) and robustness (e.g. speed of response). That is, if speed is of the essence, proteins are all premade and ready to go, and controlled allosterically and covalently modified, on millisecond timescales. At the other extreme, just-in-time transcriptional control can make new proteins on demand, but this takes orders of magnitude longer but is also much more efficient. On even longer time scales, cells rewrite their genomes by

HGT and various other mechanisms. What is new and intriguing is the extent to which control at the RNA level is an evolutionary sweet spot for balancing speed and efficiency.

This is all worked out and doesn't need anything new really, but will take some time to tell. What is unresolved is what "universal laws" exist that constrain these tradeoffs beyond what is in Chandra et al. But it's the big Turing-like result in biology (at the bacterial level), but will have to wait on a long review … (though *Nature* has never published anything remotely like this, and *Science* is just starting to, and is encouraging this direction, but we'll see….)

Aside on proteins: We don't know in what sense proteins might be universal machines. That is, is there 1) a class of reactions that have universal laws in terms of hard limits on how fast, efficiently, and robustly they can be catalyzed? And are 2)-3) proteins universally able to catalyze these reactions? This would be all 3 steps of the Turing program applied to proteins. I have no idea how to this but an essential first step seems to be to develop a full nonequilibrium thermodynamics and statistical mechanics (physics lacks this despite the existence of the terms), which seems doable and is in progress (see *IEEE TAC* paper by Sandberg, Delvenne, and Doyle)

A great starting reference on all this HGT and other mechanisms of natural genetic engineering (but that is oblivious to the whole architecture and Turing angle, but lays out the facts) is

- Shapiro, J.A. 2011. *Evolution: A View from the 21st Century*. FT Press Science

See also the recent papers above on architecture for some discussion of this, and earlier some high level commentary:

- M. E. Csete and J. C. Doyle. Reverse engineering of biological complexity. *Science* 295(5560):1664-1669 (2002).
- Csete M.E. and J.C. Doyle, 2004, Bow ties, metabolism, and disease, *Trends in Biotechnology*, Vol 22, Issue 9, pg. 446-450
- Doyle J, and Csete M, Rules of engagement. *NATURE* 446 (7138): 860-860 APR 19 2007

The biology last punch line, the 1952 paper

What about the 1952 paper? It has been systematically misread, so let's see if we can put it into the context of Turing's overall approach.

I agree that what the embryology types (Sydney Brenner, Lewis Wolpert, Jani Nusslein-Wolhard, Eric Wieschaus) imply when they respond to 'emergence' in horror is something related to the above discussion on architecture, modularity, nonlinear dynamics, etc and is indeed first connected to the 1936 paper, but the 1952 paper took an essential step, and I'll try to explain what that essential step was (and pattern formation per se was not the essential element).

So I want to finish this story with the 1952 paper and explain that many of the narrow interpretations of Turing patterns are now clearly irrelevant to most aspects of development, and this takes many steps, and not sure how much we'll have space for… but here goes…

The first misreading of Turing was to try to put it within the emergence camp (SPCS, e.g. nonlinear dynamics = chaos) whose goal is essentially to explain patterns in nature (fractal mountains and shore lines and lung branching, leaf shapes, camouflage, etc) in terms of simple models inspired by statistical physics. Turing can certainly be read as epitomizing "emergence", in that Turing patterns (and hopefully eventually the stat phys holy grail of chaos and power laws) can be created with simple fairly homogeneous ensembles of interacting chemical species described by simple nonlinear equations.

It's very unlikely that patterns per se were ever Turing's ultimate goal and it is now clear that for real biology they are never ends in themselves (except perhaps in camouflage) but are intermediate readouts of components in complex control systems. Turing knew, even if he never stated it explicitly, that in order to build robust predictable repeatable machines that are also flexible, adaptive, and evolvable (and development is a striking example of such machines, as is the whole bacterial biosphere) you needed a physical substrate that could be made into 'discrete state machines' or something equivalent. Since biologists tend to talk about cell types as consisting of discrete types, even if this is an approximation, Turing would have been naturally looking for ways that chemistry (rather than electronics) could implement the essential digital nature that underlies Turing machines.

I think if he were to have lived he might have shifted his focus to bacteria and internal cell control mechanisms, and the role of the obviously digital nature of DNA and RNA and protein sequence, in the ultimately analog control of the cell… but in 1952 the most obvious question for Turing was how could morphogenesis implement a very rudimentary kind of machine. So Turing was doing some speculative reverse engineering based on a very abstract view of biochemistry, with few facts to go on, and what he accomplished was, particularly with our modern view, pretty astonishing. That it has been so badly misread is sad but not surprising. Almost everything that touches on architecture is wrong.

And I hope Turing would have been happy with the layered architectural view of the cell sketched above. We now have the facts and biochemical mechanisms to flesh out what sort of UTM the cell actually is. But there's more…

… the idea of morphogens as the communication method between cells which are implementing layered virtualized Turing-like universal architectures fits the latest developments in development perfectly. Turing would have naturally come to this eventually as the molecular details unfolded, and likely would have helped avoid the many cul de sacs along the way. We now understand that it is the dynamics of morphogen signaling and complex feedbacks with cell surfaces and internal states that is often important, exactly as would necessarily be true of any robust communication and control networks between such "universal machines" whether implemented electronically, mechanically, or chemically.

Also, the 1952 paper is so modern in its blend of theory and computation, and there is less like it between that paper and around 2000, when an interest in dynamics, control, mechanism, etc, resurfaced. There was a huge effort in physiology a la Guyton until 1970s, but after that it was all genomics and computation was mostly devoted to bioinformatics.

The whole EvoDevo story also fits perfectly, though the situation is much more complex and the details are not yet as clear as in the bacterial biosphere and the Internet, where the interplay of robustness, evolvability, and architecture can be explained (finally) in fair detail… Though obviously for this venue they will have to be summarized.

Many development papers are relevant here, but most important is Kirschner and Gerhart's work. G&K are referenced heavily in my papers and I need to extract the relevant story about architecture as "constraints that deconstrain" that is perfectly compatible with Turing but hasn't been explained that way.

So the bottom line is that the relevance of the 1952 paper can only be really understood in the context of 1936, the cyberwar part of WW2, and Turing's design of the real general purpose computers. It has been systematically misread and the reactions of the embryologists all along have been instinctively right but would have helped greatly to have been informed by Turing's likely real intent.

What I'm leaving out so far but would like to put in eventually in another venue, the 1950 Turing test paper

`A big piece of the overall story that is really rich but I think has to be triaged in this pass is the connections this all has with the 1950 Turing test paper. The point to be made is that Turing focused on what systems *did*, their external behavior, and recognized that there might be lots of internal implementations that yielded the same behavior (the essence of his universality, though here virtualization and layering capture the essence) and we might get distracted from the essence of this by focusing on internal states too much. The Turing test has generated perhaps the largest popular literature of anything he has done, and it seems still misunderstood (like the 1952 paper) because it isn't adequately grounded in understanding of his whole approach to systems. It is also clear he is being deliberately whimsical and provocative, to stir up discussion, and the literature since proves he was successful.

One striking observation is Turing's whole description of the human mind/brain as a layered onion

> "The 'skin of an onion' analogy is also helpful. In considering the functions of the mind or the brain we find certain operations which we can explain in purely mechanical terms. This we say does not correspond to the real mind: it is a sort of skin which we must strip off if we are to find the real mind. But then in what remains we find a further skin to be stripped off, and so on. Proceeding in this way do we ever come to the 'real' mind, or do we eventually come to the skin which has nothing in it? In the latter case the whole mind is mechanical."

Wow. It's hard to find anything that is a pithier summary of the modern cognitive neuroscience view of the brain\mind. See my 2011 *PNAS* paper for more details.

This is a huge piece of the "architecture" story, but is hard to fit in with between the 1936 and 1952 papers, and not as urgent for me (given the 2011 *PNAS* paper that gives a start, but doesn't make the ultimate connections) as the layered bacterial biosphere architecture piece or the modern dynamic morphogenesis story.

The much bigger picture not included at all

In summary: This new mathematical frameworks for the study of complex networks (that Turing's 1936 paper was arguably the canonical precursor) suggests that all this apparent network-level evolutionary convergence within/between biology/technology is not accidental, but follows necessarily from their universal system (and hard limits) requirements to be fast, efficient, adaptive, evolvable, and most importantly, robust to perturbations in their environment and component parts. The universal hard limits on systems and their components have until recently been studied separately in fragmented domains of physics, chemistry, biology, communications, computation, and control, but a unified theory is needed and appears feasible.

We have both the beginnings of the underlying mathematical framework and also a series of case studies in classical problems in complexity from statistical mechanics[28], turbulence[27][32] , cell biology

[1][3][5][6][7][10][14][19][20][23][29][30], neuroscience[31], wildfire ecology[17][25], earthquakes[33], and the Internet [4][8][9][11][15][18][21].

Selected references:

[1] Yi TM, Huang Y, Simon MI, Doyle J (2000) Robust perfect adaptation in bacterial chemotaxis through integral feedback control. *Proc Natl Acad Sci* USA 97:4649-4653.

[2] Carlson JM, Doyle J, Complexity and robustness, *P Natl Acad Sci* USA 99: 2538-2545 Suppl. 1 FEB 19 2002

[3] M. E. Csete and J. C. Doyle. Reverse engineering of biological complexity. *Science* 295(5560):1664-1669 (2002).

[4] Low SH, Paganini F, Doyle JC, Internet congestion control, *IEEE Contr Syst Mag* 22 (1): 28-43 FEB 2002

[5] Csete M.E. and J.C. Doyle, 2004, Bow ties, metabolism, and disease, *Trends in Biotechnology*, Vol 22, Issue 9, pg. 446-450

[6] J. Stelling, U. Sauer, Z. Szallasi, F. J. Doyle III, and J. Doyle, 2004, Robustness of cellular functions, *Cell*, October, 2004.

[7] H. El-Samad, H. Kurata, J.C. Doyle , C.A. Gross, and M. Khammash, (2005), Surviving Heat Shock: Control Strategies for Robustness and Performance, *PNAS* 102(8): FEB 22, 2005

[8] Jin C, Wei D, Low SH, Bunn J, Choe HD, Doyle JC,et al (2005), FAST TCP: From theory to experiments *IEEE NETWORK* 19 (1): 4-11 JAN-FEB 2005

[9] Paganini F, Wang ZK, Doyle JC, et al. Congestion control for high performance, stability, and fairness in general networks , IEEE-ACM TRANSACTIONS ON NETWORKING 13 (1): 43-56 FEB 2005

[10] J. Doyle and M. Csete (2005). Motifs, stability, and control. *PLOS Biology*, 3, 2005.

[11] Wang JT, Li L, Low SH, Doyle JC. (2005) Cross-layer optimization in TCP/IP networks, *IEEE-ACM TRANS ON NETWORKING* 13 (3): 582-595 JUN 2005

[12] T Brookings, JM Carlson, and J Doyle (2005) Three mechanisms for power laws on the Cayley tree, Phys. Rev. E 72, 056120

[13] Manning M, Carlson JM, Doyle J (2005) Highly optimized tolerance and power laws in dense and sparse resource regimes PHYSICAL REVIEW E 72 (1): Art. No. 016108 Part 2 JUL 2005

[14] R. Tanaka, T-M Yi, and J. Doyle (2005) Some protein interaction data do not exhibit power law statistics, FEBS letters, 579 (23): 5140-5144 SEP 26 2005

[15] Doyle et al, (2005), The “Robust Yet Fragile” Nature of the Internet, *PNAS* 102 (41), October 11, 2005

[16] Zhou T, Carlson JM, Doyle J (2005) Evolutionary dynamics and highly optimized tolerance, *JOURNAL OF THEORETICAL BIOLOGY* 236 (4): 438-447 OCT 21 2005

[17] MA Moritz, ME Morais, LA Summerell, JM Carlson, J Doyle (2005) Wildfires, complexity, and highly optimized tolerance, *PNAS*, 102 (50) December 13, 2005; ,

[18] L Li, D Alderson, JC Doyle, W Willinger (2006) Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications, *Internet Math*, Vol. 2, No. 4, 2006

[19] H El-Samad , A Papachristodoulou, S Prajna, J Doyle, and M Khammash (2006), Advanced Methods and Algorithms for Biological Networks Analysis, *PROCEEDINGS OF THE IEEE*, 94 (4): 832-853 APR 2006

[20] Kurata, H El-Samad, R Iwasaki, H Ohtake, JC Doyle, et al. (2006) Module-based analysis of robustness tradeoffs in the heat shock response system. *PLoS Comput Biol* 2(7): July 2006

[21] M Chiang, SH Low, AR Calderbank, JC. Doyle (2006) Layering As Optimization Decomposition, *PROCEEDINGS OF THE IEEE*, Volume: 95 Issue: 1 Jan 2007

[22] Martins NC, Dahleh MA, Doyle JC (2007) Fundamental Limitations of Disturbance Attenuation in the Presence of Side Information, *IEEE Trans Auto Control*, Feb 2007

[23] Doyle J, and Csete M, Rules of engagement. *NATURE* 446 (7138): 860-860 APR 19 2007 (PMID: 17443168)

[24] Willinger W, Alderson D, and Doyle JC (2009) Mathematics and the internet: A source of enormous confusion and great potential*. Notices Amer Math Soc* 56:586-599.

[25] Bowman, Balch, Artaxo, Bond, Carlson, Cochrane, D’Antonio, DeFries, Doyle, Harrison, Johnston,. Keeley, Krawchuk, Kull, Marston, Moritz, Prentice, Roos, Scott, Swetnam, van der Werf, Pyne (2009) Fire in the Earth System, *Science* 24 April 2009: **324** (5926), 481-484.

[26] Alderson DL, Doyle JC (2010) Contrasting views of complexity and their implications for network-centric infrastructures. *IEEE Trans Systems Man Cybernetics—Part A: Syst Humans* 40:839-852.

[27] Gayme DF, McKeon BJ, Papachristodoulou P, Bamieh B, Doyle JC (2010) A streamwise constant model of turbulence in plane Couette flow*, J Fluid Mech*, vol 665, pp 99-119

[28] H. Sandberg, J. C. Delvenne, J. C. Doyle. On Lossless Approximations, the Fluctuation-Dissipation Theorem, and Limitations of Measurements*, IEEE Trans Auto Control*, Feb 2011

[29] G. Buzi, U. Topcu, J. Doyle. Analysis of autocatalytic networks in biology, *Automatica*, In Press, Available online 21 March 2011 http://dx.doi.org/10.1016/j.automatica.2011.02.040

[30] Chandra F, Buzi G, Doyle JC (2011) Glycolytic oscillations and limits on robust efficiency. *Science*, Vol 333, pp 187-192.

[31] JC Doyle, ME Csete (2011) Architecture, Constraints, and Behavior, *P Natl Acad Sci USA*, vol. 108, Sup 3 15624-15630

[32] Gayme DF, McKeon BJ, Bamieh B, Papachristodoulou P, Doyle JC (2011) Amplification and Nonlinear Mechanisms in Plane Couette Flo*w, Physics of Fluids*, in press (published online 17 June 2011)

[33] Page, M. T., D. Alderson, and J. Doyle (2011), The magnitude distribution of earthquakes near Southern California faults, *J. Geophys. Res.*, 116, B12309, doi:10.1029/2010JB007933.