**Introduction**

This paper aims to explore the properties of an important subset of technical artifact, tooling. I define tooling loosely as the set of all technical artifacts designed and deployed explicitly in the service of manufacturing new artifacts. Conventional examples of tooling are machine tools such as the engine lathe and vertical milling machine, and cutters such as endmills and carbide inserts. In the context of a factory or workshop, tooling is the set of objects directly *necessary* for the manufacture of whatever is being manufactured.In this paper I seek to identify and characterize the attributes that make a technical instrument tooling, and argue that tooling, like scientific instruments, is a discrete type of technical artifact, identifiable by both its physical properties and its function. This paper is divided into two parts. The first examines the functional properties of traditional examples of conventional tooling, such as machine tools. The second part extends this examination to computational systems and argues that general-purpose computing devices such as CPUs and GPUs are themselves general-purpose tooling.

**Tooling**

Within the engineering, manufacturing and production disciplines, “tooling” is a jargon term referring to the collective machines, templates, molds, jigs, fixtures, and other equipment used to manufacture, assemble, or process products, components, or materials in an industrial setting. Like all instruments, tooling is mind dependent (Houkes & Vermass, 2014); and I argue that holds particular epistemic significance in that its goal is to make material the idealized intentions of the designers using it. Tooling is the mechanism by which humans attempt to translate idealized[[1]](#footnote-1) knowledge into material reality.[[2]](#footnote-2) High precision machining allows designers to realize their designs at a level of accuracy indiscernible from perfection when viewed at the macroscopic scale. This property gives tooling a special epistemic significance among instrument kinds; it is the technology that translates the ideal into the real.

**From Tools to Tooling**

*Tool use and development is fundamentally coupled with human knowledge systems. Tools are integral to human progress because they both allow effective exploitation of the environment and foster efficient transfer of knowledge between people.*

While considering the statement above, I looked for examples of tools that don’t carry epistemic cargo and concluded that the marriage of exploitation systems and knowledge systems is an integral property of tools and does well to characterize the evolution of tools throughout human civilization, and the evolution from tools to tooling. For example, the stone axe is a tool that does not necessarily require elaborate knowledge of its possible and “appropriate” applications to be effectively used; it is likely that a human user will quickly learn that the stone axe can be used to enhance abilities already present in their bodies, namely bashing and tearing[[3]](#footnote-3). While transfer of knowledge from expert users should improve a novice user’s capabilities with a stone axe, the tool does not require much epistemic infrastructure.

As tools developed, so did the epistemic infrastructure supporting them. Consider the tools of neolithic agriculture, many of which were essentially advanced implementations of the stone axe fashioned to achieve specific agricultural tasks such as plowing, sowing and reaping (Fowler et al., 2014). These tools not only required training and understanding to use effectively, but importantly, each tool was a component in a larger system of tools required to carry out a large-scale epistemically complex operation.

Skipping ahead 10,000 years or so, let us consider the screwdriver. While operation of a screwdriver may not require integration of complex knowledge, the tool system which it belongs to (threaded fasteners) is a vast constellation of precisely manufactured, application-specific tools with an epistemic infrastructure that is only fully comprehended by expert engineers and machinists.

The examples above are intended to illustrate that the evolution of tools and tool use can be characterized in terms of an evolution from embodied interactions to epistemic interactions. The primary function of all tools is to abstract away the weakness and imprecision of the body while sharpening the precision of epistemic-to-material translation. In this sense, tools have always been progressing toward tooling, and it appears that this process of automation will continue ad infinitum. The total automation of physical labor is already a reality in some contexts, and the automation of epistemic labor increasingly so.

While the precise boundaries of the industrial age are fuzzy, I believe that the primary force that drove it was the transition from tools to tooling. This transition happened gradually and by degree[[4]](#footnote-4), but at the point that the human body is abstracted out of the physical control loop dictating a given tool’s operation (the tool gains the property of autonomy), tooling and hand tools became *different kinds* of instruments. In support of this argument, I have selected four properties that I believe specifically identify a technical artifact as tooling: precision, generality, autonomy, and productivity. The following equation describes tooling in set builder notation, and each property is described on the following page.

Equation 1: Let T denote the set of all tooling, where each technical artifact t is an element of U, the universal set of technical artifacts. A technical artifact t is included in T if and only if it possesses the properties of precision (P1), generality (G), autonomy (A), and productivity (P2).

*Precision:* Tooling is not used for measurement, but it is produced according to precise measurements[[5]](#footnote-5). Precision is the critical attribute that makes mass production possible and enables strictly prescribed make plans.

*Generality:* Tooling of all scales, from a single endmill to a 5-axis machining center, is designed with replaceability and generality in mind. Tooling systems are designed within highly specific structural and functional parameters, but the intended use of a tooling system is not strictly defined by its designers. While some tooling is more general than others, all tooling systems[[6]](#footnote-6) are designed to accommodate a range of applications.

*Autonomy:* Tooling operates independently of direct operator control. While an operator may be involved in the assignment of a tooling task, they are never in direct contact with the with tool or exerting free control over it. By this definition, hand tools are not tooling.

*Productivity:* Whether by additive, reductive or formative processes, tooling produces changes in the material substrates it works on. Many types of artifacts may be involved in the manufacture of products, but if they do not produce changes in the material substrate being manipulated, they are not tooling.

While each of these properties were present in tooling that existed prior to the first industrial revolution (e.g. the printing press), asynchronous advancements in each property were involved in sparking the next three industrial revolutions. Broadly speaking, advances in generality enabled mass production and sparked the Technological Revolution, advances in autonomy sparked the Digital Revolution, and advances in productivity, particularly in the realm of data manipulation, sparked the current industrial revolution. Across each of the four industrial revolutions, the precision of tooling and tooling techniques has increased dramatically, with machining tolerances shrinking by about two orders of magnitude (Roe, 1916; Gresik, 2016) and present-day microfabrication technologies such as EUV photolithography and two-photon 3d printing operating at the sub-micron scale (Peeters et al., 2017; Geng et al., 2019).

**On Tooling as an instrument subset**

In the interest of exploring the ontological status of tooling, I have attempted to situate it within the ontology of system of instruments described in Houkes & Vermaas (2014). While only hinting at the material realities involved, the authors do well to recognize the modern state of production design and manufacturing. They acknowledge that design and manufacturing are discrete activities characterized by both the division of labor and division of intentions between designers and makers. They define their system of technical instruments in terms of use plans (*up*) proposed by designers, goals (*gu*), and goal-contributing capacities (*φ*). They then proceed to describe a comprehensive, if vague, ontology of instrument subclassification primarily operating under these terms. Under the Houkes & Vermaas classifications, tooling would fall under each of the three instrument subclasses, depending on the degree of generality (*G(t)*) present in the tooling. Below, I’ve adapted Table 10.1 from Houkes & Vermaas (2014) to illustrate where various examples of tooling fall within their instrument system, and how their classification relates to the property of instrument generality (*G(t)*) described in Equation 1. In each set of examples, I have listed tooling from traditional manufacturing and production environments, along with examples of computational tooling. I will provide more detail on the computational examples later in the paper.

|  |  |  |  |
| --- | --- | --- | --- |
| Table 1. Tooling as classified within the Houkes & Vermaas instrument system. | | | |
| **Instrument subclass** | **Description** | **Tooling Examples** | **Generality** |
| *gu* goal instruments | items intended by designers to be interacted with in use plans with goal *gu* | injection molds, application-specific integrated circuits (ASICs) | *G(t) = low* |
| *up* plan instruments | items intended by designers as items to be interacted with in use plan *up* | milling machines, BIOS chips | *G(t) = medium* |
| *φ* functional instruments | items intended by designers as items to be interacted with in use plans for capacity *φ* | CPUs, GPUs, endmills, lathe tooling | *G(t) = high* |

In considering the mapping between the Houkes & Vermaas instrument system and my definition of the tooling subset of technical artifacts, I was confronted with the possibility that tooling might not be “special,” as it fell relatively neatly into their more general instrument classification system. As their 2014 article is focused on situating technical artifacts in the ontology of natural kinds, the breadth and vagueness definition of their definition worked elegantly and made plain to me that I am approaching the discussion of tooling from a relatively practical (crude) perspective. As I continue to develop the ideas in this paper, I hope to better situate tooling within the philosophy of technology, but the goal at hand is to get my current ideas on machine and digital tooling down on paper.

General-purpose tooling is not *designed* in the sense proposed by Houkes & Vermaas, it is engineered. It is manufactured in service of the physical properties it possesses, but the capacities that these properties afford are not dictated, they are for the user to decide. Even in the case of *G(t) = medium* tooling, functional capacities are extremely broad. *G(t) = medium* tooling is general purpose, but qualifies as a plan instrument because it requires a plan[[7]](#footnote-7) to execute, and these plans have finite degrees of freedom. General purpose tooling is any tooling capable of uses that its designers never dictated. When I first conceived of the topic of this paper, I was thinking of all tooling as general purpose tooling. As I developed the ideas and did background reading, I realized that generality is a variable property, and that specialized tooling may be a *different kind* of technical instrument than general-purpose tooling. As discussed on page two, I see the transition from specificity to generality as a marker of the transition from tools to tooling; two different kinds of instruments. It may be that the term “specialized tooling” is paradoxical under my definition, and that the definition is that of general-purpose tooling.

During my very brief study of the philosophy of technology, I’ve frequently wondered if it is petty to worry about narrow subclassifications of things; it seems that the most elegant propositions are simple, grand, and to some degree timeless. As I’ve quarreled with this thought, I’ve come to decide that tooling is not some mere special interest, but a fundamentally important artifact kind in that it breathes life into technology. While tooling is certainly a mind-dependent instrument in most cases, it is possible that we are nearing a point where the autonomy of tooling obviates human intervention, and tooling becomes fully (human) mind-independent[[8]](#footnote-8). This is the promise and threat of abstracting away our mental and physical work to advanced information systems.

**Control Systems**

Before I introduce the argument that computers are tooling, I want to take a moment to point out that the motivations that drove machine and computer technological development are largely the same. Revisiting my definition of the set of all tooling, a latent property emerges out of the four requisite properties (precision, generality, autonomy, and productivity) is *control*. Improvements in any of the four properties result in a higher agency[[9]](#footnote-9) system. Control weaves a thread through all advancements in technology, from stone axes to steam engines to computers. While the connection between heavy machinery and computers may not seem obvious at first, these technologies are deeply connected from the perspective of control. At its simplest, this control can be observed in the spatial precision of machine tools and the repeatable accuracy[[10]](#footnote-10) of computational outputs. Machine tools are characterized by control of translational movement, their movement along a chosen axis is tightly controlled, and movement out of parallel with that axis is restricted. Additionally, automated, closed-loop control systems have featured heavily in machine design since before the first industrial revolution. The centrifugal governor, a closed-loop control system made famous in association with the Watt Steam Engine, was invented in the 17th century for the regulation of water wheels and windmills (Bellman, 2015). The centrifugal governor was a sort of mechanical computer, which produced reliable and repeatable outputs in response to its inputs. This sort of reliability was one of the chief motivations for building computers, well before it was realized that a computer could operate as a universal Turing Machine.

Control systems and machine precision have advanced in concert with one another, as they were developed in pursuit of many of the same goals. While effective machine tools predate effective computer systems, they share a common evolutionary path. In many ways, these two fields of technology address complementary aspects of the same tooling property, autonomy. Where machine tooling abstracts away *physical* responsibilities from the human operator, computational tooling abstracts away *mental* responsibilities from the operator. It should be noted that the first computer numeric controlled (CNC) machine went into operation in 1952. Far before the development and deployment of desktop PCs in offices and homes, computer-controlled machine tools were an indispensable part of the precision manufacturing ecosystem.

**Computers Are General-Purpose Tooling**

Perhaps ironically, the technical bottleneck that hamstrung full-scale development of Charles Babbage’s difference engines was that the metalworking techniques of the era could not economically produce the parts at the precision required, which eventually lead to the British government abandoning funding of the project (Babbage, 1864). While the Babbage engine was not the first programmable computational device, it was the first general-purpose computer design in the sense that it was Turing complete, automatic, and explicitly designed to automate a broad domain of mental labor performed by humans (mathematical calculation). I will spare the reader a detailed history of early computers, but I mention the Babbage engine to demonstrate that the goal of computational systems has long been to automate mental work that would otherwise be performed by humans, in the context of *general-purpose use plans*.

While early computers were deployed to perform traditionally mathematical tasks, by the mid-20th century it was well understood that a Turing-complete computer could theoretically compute any computable algorithm, and therefore not limited to calculation. By the 1950s, computers were already being applied to problems outside of traditionally mathematical domains. For example, the Georgetown-IBM experiment of 1954 demonstrated Russian to English text translation using an IBM 701 computer (Hutchins, 2004).

General purpose computers are like factories without make plans. The power of the computer is its capacity to execute any make plan describable by the designers (e.g. software engineers) using it. Computer engineers seek to maximize the computational throughput of a system, within the boundaries dictated by practical and economic feasibility; but the goal of their use plan is to manufacture a tooling system that kernel engineers and software engineers can use to implement use plans and make plans of their own. Like general-purpose machine tooling, computer hardware is not designed in the service of use plans, but in the service of broad functionality. The product design and manufacturing design aspect of computers is addressed in software. From the frame of reference of the end-user, software serves as both the make plan and use plan in a computational system. A computer without software running on it is a collection of tooling waiting for a make plan, A dormant factory stacked to the brim with general-purpose tooling.

While computer hardware is manufactured according to unfathomably precise and complicated make plans, it is designed with maximally unbounded use plans. This is the dual nature of all general-purpose tooling: its make plan is maximally specific[[11]](#footnote-11), its use plan is maximally broad. General-purpose computers are general-purpose tooling, and in fact they are more general than any tooling used in traditional manufacturing settings.

**On Dedicated and Flexible Computational Tooling**

Early in the development of this paper, I considered generality a required property of tooling. After some consideration, I termed this property “generality,” but stated the caveat that while *tooling systems* are general purpose, some tooling is developed for singular, specialized applications. For example, injection molding tooling, specifically the molds themselves, are not general-purpose. Injection molds are clearly a type of tooling. They are designed and deployed in the service of production. However, they are not designed with reconfiguration or flexibility in mind. In fact, they have very tight item definitions and use plans, and are mostly useless outside of their primary function. Injections molds are an example of what I term “dedicated tooling.” Tooling which is application specific to the production of a single product. Other examples of dedicated tooling include product-specific jigs and fixtures, cutting dies, and application-specific integrated circuits (ASICs). As main the objective of this paper is to argue that computers are digital tooling, I think it is necessary to explore the degree to which they are general-purpose or not. As stated earlier, I am confident that CPUs are general-purpose tooling; in fact, general-purpose computation is the most stable property across CPUs of all eras and instruction set architectures.[[12]](#footnote-12) In the common vernacular, “computer” refers to a machine running on a Von Neumann architecture, with a CPU, RAM and so on. However, not all computational devices are governed by the operation of CPUs. In the following paragraphs, I will present a few case studies on the relative tooling-flexibility of several integrated circuit architectures.

*555 Timer IC:* Integrated circuits tend to be flexible by design. In most cases, they are designed with modularity in mind. For example, the “555 timer” (1972) is an 8-pin integrated circuit that can function either as a timer, delay, pulse generator, oscillator or flip-flop simply by manipulating the current entering its inputs. The 555 timer has a defined set of capabilities but was designed to work in a variety of configurations to serve a variety of purposes. The 555 timer is not a highly flexible tool, neither is it dedicated. CPUs and other Turing-complete hardware architectures are examples of highly flexible digital tooling.

*ASIC:* By contrast, an Application-Specific Integrated Circuit (ASIC) is characterized by a strict use plan and a lack of flexibility. As with injection molds, lack of flexibility is justified by efficiency. For example, a BitMain Antminer ASIC (2018), designed and deployed specifically to mine Bitcoin, is vastly faster and more energy efficient at mining Bitcoin than a comparably-priced GPU, however it cannot drive a Turing-complete computer. An Antminer has no capacity to help its user make art, conduct science, or socialize online. ASICs such as the Antminer are *dedicated digital tooling*.

*NorthPole:* NorthPole is a next-generation neural inference architecture being developed by IBM Research (Modha et al., 2023). It has a variety of energy-saving features that make it an attractive alternative to GPUs for AI applications, but in service of the current discussion I will focus on one: *it is inference-only*. The NorthPole chip does not train neural networks, nor can it. It cannot run general purpose software or serve as the processor for a general-purpose computer. However, it can run inference on any neural network that can fit in its memory. NorthPole essentially becomes a hardware instantiation of the neural network that is loaded on it. In this sense, NorthPole differs from an ASIC in that it could be used in a huge range of applications, though it is nowhere nearly as flexible as a CPU. Northpole, and neural inference chips in general, are notable because we are likely entering an era in which the “intelligent” digital devices we interact with are no longer powered by general-purpose computers. NorthPole is an example of digital tooling that is dedicated and flexible at once.

FPGAs: Field Programmable Gate Arrays (FPGAs) are integrated circuits that are designed to be configured by the end-user “in the field”. FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together to fit whatever intention the user desires. For example, an FPGA could be programmed to fit a highly specific application, such as military-grade signal processing, or it could be programmed into a fully functioning general-purpose CPU. FPGAs are a highly flexible subclass of digital tooling. Because they are essentially programmable hardware, FPGA hardware architectures can be changed on the fly, making them robust against hackers as it is difficult to exploit a system when you don’t know what system you’re exploiting. This property makes them especially useful for military and aerospace applications (Rockett et al., 2007), but they are also commonly used in civilian industry to prototype new hardware architectures.

Within the class of all tooling, computational or machine, there is general-purpose and special-purpose tooling. I believe that the current thrust of technological development favors general-purpose tooling. A potential example of this would be the shift away from injection molding toward 3d printing. Technologies such as generative AI and 3D printing, and even targeted online advertising, show the possibility of a turn away from mass production in favor of mass customization.

**Conclusion**

In this paper, I made an initial exploration into the properties that define tooling as a distinct subset of technical artifact. Through an analysis of traditional examples of tooling, I have identified four key attributes: precision, generality, autonomy, and productivity. These properties collectively distinguish tooling from other types of instruments and support its unique role in translating idealized intention into material reality. The evolution of tools throughout human history can be characterized by a progression from embodied interactions to epistemic interactions. As tools developed, they became increasingly reliant on complex knowledge systems and required greater expertise to operate effectively. The transition from tools to tooling has coupled with the development of human civilization, with tooling abstracting away the limitations of the human body and enabling greater precision in the realization of product designs.

The main objective of this paper was to argue that general-purpose computing devices, such as CPUs and GPUs, can be considered a form of digital tooling. Like machine tooling, computers are designed to automate and abstract away human labor and provide a flexible system of production to meet a wide range of goals. And, as with machine tools, not all computational devices exhibit the same degree of generality. While CPUs are highly flexible and can be considered general-purpose tooling, other architectures, such as ASICs and neural inference chips, have more constrained functionality. It is entirely possible that CPUs represent a sort of “wild west” of computing, and that at some point, general-purpose computers will once[[13]](#footnote-13) again become an instrument used by specialists.

In situating tooling within the system of instruments proposed by Houkes and Vermaas, I was surprised to see how neatly the degree of generality *G(t)* influences the classification of tooling as a goal, plan, or functional instrument. This analysis revealed the importance of generality in defining the nature of tooling, and led me to question aspects of my definition of tooling. It is possible that what I have termed “dedicated tooling” is not tooling at all under the definition I proposed. As I rework this paper, it may be worth it to let go of the industrial terminology to more directly address the technological property I’m most interested in, generality of function.

**References:**

Houkes, Wybo, and Pieter Vermaas. “On what is made: instruments, products and natural kinds of artefacts” Springer (2014)

Babbage, Charles “Passages from the Life of a Philosopher.”

*Project Gutenberg* (November 9, 2020)

<https://www.gutenberg.org/files/57532/57532-h/57532-h.htm>

Modha, Dharmendra S., Filipp Akopyan, Alexander Andreopoulos, Rathinakumar Appuswamy, John V. Arthur, Andrew S. Cassidy, Pallab Datta, et al. “Neural Inference at the Frontier of Energy, Space, and Time.” *Science* 382, no. 6668 (October 20, 2023): 329–35. <https://doi.org/10.1126/science.adh1174>.

Pérez, Jorge, Javier Marinković, and Pablo Barceló. “On the Turing Completeness of Modern Neural Network Architectures.” arXiv, January 10, 2019. <http://arxiv.org/abs/1901.03429>.

Bellman, Richard E. “Adaptive Control Processes, A Guided Tour”

Princeton University Press (2015)

Hutchins, W. John. “The Georgetown-IBM Experiment Demonstrated in January 1954.” In *Machine Translation: From Real Users to Research*, edited by Robert E. Frederking and Kathryn B. Taylor, 3265:102–14. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. <https://doi.org/10.1007/978-3-540-30194-3_12>.

Rockett, Leonard, Dinu Patel, Steven Danziger, Brian Cronquist, and J.J. Wang. “Radiation Hardened FPGA Technology for Space Applications.” In *2007 IEEE Aerospace Conference*, 1–7. Big Sky, MT, USA: IEEE, 2007. <https://doi.org/10.1109/AERO.2007.353098>.

1. While many tool systems translate ideas into reality, for instance those used to construct architecture, only in the realm of precision manufacturing are tolerances fine enough to consider the possibility that the idealized form is made material. In disciplines such as architecture, lack of precision is a necessary evil. [↑](#footnote-ref-1)
2. One might say that artists translate mental idealizations to reality, but I argue that art is seldom done in the attempt to translate a perfect idealization of form into the material world. When this has been attempted in art, as with minimalism, the tools employed tend to be the same as those used in industrial production. [↑](#footnote-ref-2)
3. At a microscopic scale, all cutting is tearing. [↑](#footnote-ref-3)
4. Of course, tools and tooling still coexist. But they are differentiated by virtue of automation. [↑](#footnote-ref-4)
5. The dimensional regularities of tooling imbue it with a sort of “thing knowledge” as described in Baird (2004). Rather than making measurements, tooling imposes measurement. –Not to say measurement systems are necessarily knowledge on their own, but they are an important component of knowledge systems. [↑](#footnote-ref-5)
6. Note that I use the term “tooling systems.” In some cases (e.g. injection molds, customized endmills) individual pieces of tooling may be specially designed for the product they are being used to manufacture. However, these tools must conform to specifications of the tooling system they integrate with. Customized tools are not designed with generality in mind, but they are general to the tooling system they are part of. [↑](#footnote-ref-6)
7. For instance a CNC machine tool requires a g-code program, which is a plan prescribing a series of precise movements. These movements are constrained by the number of axes of travel present in the machine tool. [↑](#footnote-ref-7)
8. In the sense that human minds conceived of and initiated technology, I could consider an argument that it may never be completely mind independent. But my personal feeling is that if a technology were sufficiently autonomous that it longer requires any human intervention to direct or select its goals, it would be mind-independent. [↑](#footnote-ref-8)
9. I intend to include a discussion of “machine agency” as I develop this paper. For the time being, let “agency” be synonymous with “control.” [↑](#footnote-ref-9)
10. While computers may not be perfectly accurate, under normal operating conditions they perform reliably and deterministically. [↑](#footnote-ref-10)
11. That is, make plans are maximally specific within the economic envelope that the engineers of the tooling are working within. [↑](#footnote-ref-11)
12. The x86 instruction set architecture (ISA) has well over a thousand instructions, unique commands that the CPU can perform. The RISC-V ISA has 47 instructions. Yet both architectures are theoretically capable of computing any computable function. [↑](#footnote-ref-12)
13. Prior to the rise of personal computers. [↑](#footnote-ref-13)