**Introduction**

This paper aims to explore the properties and ontological status of an important subset of technical artifact, tooling artifacts. I define tooling as the set of all technical artifacts designed and deployed explicitly in the service of manufacturing new artifacts. 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 characterize the identifying attributes of 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 structural and 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 tooling artifacts.

I will first provide a conventional description of tooling, then situate it within the framework of technical artifacts as described in Kroes (2010) and Houkes & Vermass (2014) using a novel definition to distinguish the tooling subset within the universal set of technical artifacts. 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 (Houkes & Vermass, 2014) all tooling is mind dependent; and I argue that holds particular epistemic significance in that it can make concrete the idealized intentions of the designers using it. Tooling is the mechanism whereby humans translate idealized knowledge into material reality.[[1]](#footnote-1) 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 drives technology and makes the imagined real.

Houkes & Vermass (2014) proposes two classes of artifacts, “instruments” and “products.” Instruments are used to achieve an intended end, whereas products are the end result of a production process. (I don’t know if I have the ideas straight enough to situate tooling in terms of their argument on natural kinds, but might see about trying). While Houkes & Vermass make a valid argument for these two types, it is fruitful to attempt to identify subsets within the instrument and product classes. Tooling clearly is a subset of the instrument class, and like scientific instruments, I believe tooling holds a separate and identifiable position in the ontology of technical artifacts.

I have selected four properties that I believe can identify an artifact as tooling: generativity, precision, generality, autonomy, and transformativity.

Precise: Tooling is not used for measurement, but it is produced according to precise measurements[[2]](#footnote-2). Precision is the critical attribute that makes mass production possible and enables strictly prescriptive make plans.­

Generic: Tooling of all scales, from a single endmill to a CNC machining center, is designed with replaceability and generality in mind. Tooling systems are designed within highly specified 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[[3]](#footnote-3) are designed to accommodate a range of applications.

Autonomous: 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.

Transformative: Whether by additive or reductive processes, tooling generates and manipulates the material substrates it is intended to work on. Many types of artifacts may be involved in the manufacture of products, but if they do not generate changes in the substrate being manipulated to achieve an end product, 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. Advances in generality enabled mass production and sparked the Technological Revolution, advances in autonomy sparked the Digital Revolution, and advances in generativity, particularly in the realm of data manipulation, sparked the current industrial revolution.

Industrial revolutions can be characterized by the production of technical artifacts that remap humankind’s relationship with nature. The principal component responsible for industrial revolutions is precision of tooling. Between periods of revolution, industrial progress realizes its technological potential within the boundaries defined by its tooling, in contrast, industrial revolutions occur in response to the expansion of these boundaries.

It’s important to note that innovation in tooling is not a discrete process, and while industrial revolutions can be described in terms of discrete states, the ideas and innovations that drive them are continuous, and precede the revolutions by decades or more.

Talk about the lapping blocks and the transistor.

Relative to Krouse:

I will argue that tooling is a subset of the universal set of technical artifacts, identifiable because its function is the production of other artifacts. –Maybe check out Krouse’s argument that technology is a life form.

Scientific Instruments:

Use the argument on scientific instruments to develop the argument of why tooling is an identifiable subset. -This should provide a good framework in terms of phrasing and terminology.

Houkes & Vermaas:

Attempt to fit tooling into their concept of an instrument system, and discuss where mind independence applies to tooling. –Tooling has properties that make it both more and less mind dependent than other kinds of instruments and products.

Definition of Tooling:

* “Tooling” refers to tools expressly developed and used for manufacturing or crafting.
* Tooling is an important subset of technical instruments.
* Technical instruments designed for broadly defined space of intentions
* The designers of tooling specify what the tools are capable of, but not what they are intended to produce.
* The scope of a given tool’s abilities are well-defined, but it is designed with a flexibility of application. Flexibility is a driving intention.
  + This could mean flexibility of function, and of configurability.

Computers as tooling:

* Like machine tools, computers are tooling.
* While they are marketed as products, each computer is a factory.
  + This goes from workstations to phones to gaming consoles, down to simple programmable logic controllers. Programmable hardware is electronic tooling, possessing a pronounced level of agency and flexibility that differentiates it from other technical artifacts.

Software as a Make Plan:

* Conventional computers are limited by the instruction sets of their processing units, but these are formal constraints rather than practical ones. A Turing-complete system can approximate any function, so a computer is only limited by its outputs and timescale.
* In the space of all possible software, we have an infinitely-extensible make plan.
  + Read back to the MP paper to get some arguments going

Hardware & Agency

* Most matter is programmable, and all tools are (the human swinging the hammer is the software) but tooling has been fashioned expressly to be programmable.
* Computers store and sculpt information. The logic gate is the cutter, memory is the fixture, instructions are the machines, and software is the make plan.
* CPUs don’t require that many degrees of freedom to maintain flexibility. x86 ISA has vastly more operations than RISC-V, and we are seeing a rise in specialized ICs that are designed to fulfill more specific intentions (e.g. tensorrent’s RISC-V hardware).
* The push towards specialization is including a lot of inference-only hardware, interesting in that it has a strictly defined capability to run software (neural networks) that are defined by their flexibility to approximate any function, and have been shown to be Turing-complete without requiring access to external memory.

**Complicating Matters: Dedicated Versus Flexible Tooling**

In this section, I'll discuss the concept of flexibility in tooling, a key characteristic that varies substantially across the class. All tooling is designed and deployed to manufacture products, but those designs range from generic to application specific.

While considering the argument that tooling is defined by a flexibility of designer intention, I ran up against a counterexample: injection molding tooling, specifically, the molds themselves. 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, use constraints and are mostly useless outside of their primary function. Injections molds are an example of what I call “dedicated tooling.” Tooling which in 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). In the following paragraphs, I will present a few case studies on the relative flexibility of several integrated circuit architectures.

555 Timer:

Integrated circuits tend to be flexible by design. In most cases, they are designed with modularity in mind. For example, the “555 timer” 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.

ASICs:

By contrast, an ASIC is characterized by application-specificity and a lack of flexibility. As with injection molds, lack of flexibility is justified by efficiency. For example, a BitMain Antminer ASIC, designed and deployed 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. One could never make art, conduct science, or socialize online using a BitMain Antimer. ASICs such as the Antminer are *dedicated* digital tooling.

NorthPole:

NorthPole is a next-generation neural inference architecture developed by IBM Research. It has a variety of energy-saving features that make it an attractive alternative to GPUs for running ANNs, but in service of the current discussion I will focus on one: it is inference-only. The NorthPole chip does not train ANNs, 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.

Conclusion

* Tooling is a distinct class of technical artifact, with a strong link to industrial revolutions.
* Tools have existed for hundreds of thousands of years, but tooling, in the fully modular and reprogrammable sense, is much newer. Possibly only as old as the lathe.
* Digital tooling has accelerated the output and development of both material and epistemic products.
* The computer has a dual nature as both product and factory.

1. One might say that art is about translating mental idealizations to reality, but I argue that art is seldom done in the attempt to realize a perfect idealization of form in the material world. When this has been attempted in art, as with minimalism, the tools employed have been the same as those used in industrial production. [↑](#footnote-ref-1)
2. The dimensional regularities of tooling imbue them with a sort of “thing knowledge” as described in Baird (2004). Rather than taking measurements, tooling demonstrates measurement. [↑](#footnote-ref-2)
3. 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 of product in mind, but they are general to the tooling system they are part of. [↑](#footnote-ref-3)