ALEC THOMPSON

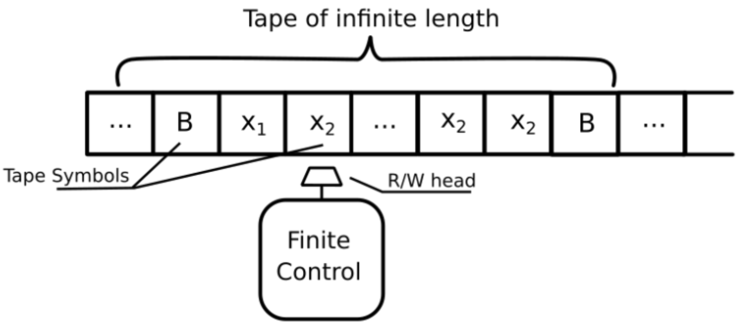
THE TURING COMPLETENESS OF LAW

1. *Introduction*

A popular question is whether the law can be replaced with a digital system capable of carrying out all legal functions. Rarely, however, do we ask the opposite: can we use a legal system to run a computer program? The purpose of this paper is to answer the second question by investigating the Turing completeness of law. It uses modern English law as a test case and constructs a Universal Turing machine (UTM) entirely out of English private law. It does so using the property rights in a piece of moveable property over time as the 'tape' of the UTM, the persons the rights are vested in as the machine's alphabet, and contract law to provide the machine instructions. Several interesting implications emerge from this thought-experiment. First, the existence of legal UTM suggests legal operations can be replicated by a digital computer and thus is relevant to the computability of law. Second, the requirements for a system to be Turing complete suggest the existence of a legal UTM in turn implies the law must possess certain properties as a system of ideas. These properties are philosophically controversial and as such the plausibility and persuasiveness of a legal UTM directly implicates jurisprudential debate. Finally, the sociological importance of legal computability is explored through a Weberian and Systems Theoretical lens, connecting the social utility of self-referential and calculable legal systems to the previous discussion. The paper is broken into three parts. Part 1 of the paper explores the concepts of Turing completeness and Universal Turing machines. Part 2 investigates the Turing completeness of law using English contract and land law as a test case. Part 3 discusses the implications of this test case in terms of jurisprudence, computability, and sociology.

2. *What is Turing Completeness?*

To understand Turing completeness it helps to understand a connected concept: the Turing machine. At its most basic, a Turing machine is an abstract machine capable of carrying out any kind of computation[1]. Turing's description of the Turing machine is very simple: it is an infinitely long piece of tape divided into cells. Each cell contains either a symbol or is blank; the total catalogue of symbols for the machine is called its 'alphabet'[2]. Above the tape is the 'head' which can read the symbol within the cell it is above. The machine can also be in different 'states' which determine how it operates. Once the head reads the cell it is positioned over, it will carry out a function according to a table of instructions and the state of the machine. The function will be some combination of moving either left or right along the tape, erasing the symbols in the cell, writing new ones, and changing machine state. For example, '**If X_1 is read whilst in state Q, erase the symbol and move to the left; then change to state P**.' Consider the diagram below[3] where the machine state and table of instructions is contained within 'finite control':



[1] For an overview of what follows, see PETZOLD C., *The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine,* New Jersey, 2008.

[2] TURING A.M., *On Computable Numbers, With an Application to the Entscheidungsproblem*, in *Proceedings of the London Mathematical Society*, 1937, p. 230.

[3] Taken from LECHENCO G., *A General Introduction to Turing Machine,* in *Open Genus,* 2018.

The tables of instructions, states, and alphabets of particular Turing machines differ, altering their functions and capacities. Nonetheless, despite the abstract Turing machine's simplicity, theoretically any calculation can be carried out by Turing machines if they are configured correctly[4]. Turing states: "*It is my contention that these operations [the operations which can be carried out using a Turing machine] include all those which are used in the computation of a number*"[5].

As noted above, particular Turing machines vary in their complexity, some being extremely basic and others more complex, depending on the size of their alphabets, number of states, and table of instructions. For example, a Turing machine can be configured such that it will carry out addition and subtraction: by feeding in tape with symbols representing the integers you want computed, the machine will print out the solution by carrying out its table of instructions. Of course, the process is extremely lengthy and complex, requiring an elaborate series of steps, including printing symbols used purely for memory purposes. Interestingly, the total configuration of the Turing machine – its table, alphabet, and states, can be reduced to an integer which Turing calls its 'description number'[6]. He then developed a particular type of Turing machine – the Universal Turing machine (UTM)[7]. If another Turing machine's description number is fed into a UTM, the UTM will duplicate the output of that Turing machine. Let us continue with our addition and subtraction Turing machine: if we input tape reading the symbolic equivalent of 4 and 3, the machine will output the equivalent of 7. Now, if we reduce our arithmetic machine to its description number and feed this into a UTM, alongside 4 and 3, the UTM will output 7 just as our original machine would. The special defining property of the UTM is that it can carry out any calculation which can be carried out on computable numbers, and as such can simulate any other Turing machine; this property can also be referred to as 'Turing completeness.' Theoretically, a Turing-complete system can run any other program or computation. The most basic UTM currently known is that of Yurii Rogozhin: it has 4 states and 6 symbols in its alphabet (4, 6), with 22

---

[4] The definition of 'calculation' is a philosophically live issue; it is, unfortunately, outside the scope of this article.

[5] TURING A.M., *On Computable Numbers, With an Application to the Entscheidungsproblem*, cit., p. 232

[6] *Ibid*, p. 240.

[7] *Ibid*, p. 241.

different instructions in its table of instructions[8]. Here is the instruction table of Rogozhin's two state (2, 18) Turing machine:[9]

q_1	$1c_2$	Lq_1	q_2	$1\overleftarrow{1}$	Rq_2
q_1	$\overrightarrow{1}\overleftarrow{1}_1$	Rq_1	q_2	$\overrightarrow{1}\overrightarrow{1}$	Rq_2
q_1	$\overleftarrow{1}c_2$	Lq_1	q_2	$\overleftarrow{1}\overrightarrow{1}$	Lq_2
q_1	$\overrightarrow{1}_1 1$	Rq_1	q_2	$\overrightarrow{1}_1\overleftarrow{1}_1$	Rq_2
q_1	$\overleftarrow{1}_1\overrightarrow{1}_1$	Lq_1	q_2	$\overleftarrow{1}_1 1$	Lq_2
q_1	$b\overleftarrow{b}$	Rq_1	q_2	bb_2	Rq_1
q_1	$\overrightarrow{b}\overleftarrow{b}_1$	Rq_1	q_2	$\overrightarrow{b}\overleftarrow{b}$	Rq_2
q_1	$\overleftarrow{b}b$	Lq_1	q_2	$\overleftarrow{b}\overrightarrow{b}$	Lq_2
q_1	$\overrightarrow{b}_1 b$	Rq_1	q_2	$\overrightarrow{b}_1\overleftarrow{b}_1$	Rq_2
q_1	$\overleftarrow{b}_1\overrightarrow{b}_1$	Lq_1	q_2	$\overleftarrow{b}_1\overrightarrow{b}$	Lq_2
q_1	b_2b_3	Lq_2	q_2	b_2b	Rq_1
q_1	$b_3\overrightarrow{b}_1$	Lq_2	q_2	$b_3\overleftarrow{b}_1$	Rq_2
q_1	$c\overrightarrow{1}$	Lq_2	q_2	$c\overleftarrow{c}$	Rq_2
q_1	$\overrightarrow{c}\overleftarrow{c}$	Rq_1	q_2	$\overrightarrow{c}\overleftarrow{c}$	Rq_2
q_1	$\overleftarrow{c}\overrightarrow{c}_1$	Lq_1	q_2	$\overleftarrow{c}\overrightarrow{c}$	Lq_2
q_1	$\overrightarrow{c}_1\overleftarrow{c}_1$	Rq_2	q_2	$\overrightarrow{c}_1 c_2$	Rq_2
q_1	$\overleftarrow{c}_1$	—	q_2	$\overleftarrow{c}_1 c_2$	Lq_1
q_1	$c_2\overleftarrow{1}$	Rq_1	q_2	c_2c	Lq_2

The two states are q_1 and q_2, the 18 different symbols are in the middle of each column, and to the right are the operations. The line q_1 $1c_2$ Lq_1 for instance states that if the head reads the symbol '1' in state q_1 then 1 is deleted and c_2 is entered in before moving to the left of the tape in state q_1. Provided a Turing machine can execute Rogozhin's machine then we be confident that the first machine is Turing complete.

3. *Is the Law Turing Complete?*

Having outlined Turing completeness, the next step is to construct a UTM out of English private law modules. This section presents two possible Universal Turing machines on the model of Rogozhin's (2, 18) UTM. The first is composed entirely out of contract terms and, as such, is highly artificial. It is unlikely a court would recognise these unusual terms or enforce them. The second is composed out of standard contract terms and property rights, arranged in a configuration that leads to the formation of a UTM. We can differentiate this UTM from the former on the basis it

[8] ROGOZHIN Y., *Small Universal Turing Machines,* in *Bulletin of Symbolic Logic,* 2003, p. 414.

[9] *Ibid,* p. 237.

is more 'organic' – it is formed using existing, recognised contract terms, albeit in an unusual configuration. Both of these systems are Turing-complete and therefore it is possible to run any computer program using them.

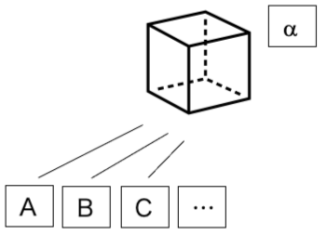
3.1. *Artificial Contract UTM*

It is straightforward to construct a UTM using contract law. One can imagine two parties agreeing a contract which incorporates all the instructions necessary for the creation of a Universal Turing machine. For example, if Party B promised Party A to draw a series of cells on a piece of paper and carry out operations according to a table of instructions contained within the contract. The terms would also include a definition of the two states, 18 symbols in the alphabet, and instructions for how to read the symbols on the tape. The halting problem can be implicated by specifying a halting state: eg. Party B is to pay *£x* in the event that the Turing machine party B is operating halts. Given the implications of the halting problem, there would be no general method to determine this in advance for any given input into the machine. The effect of this contract, although incorporated in extremely clear terms, could therefore be impossible to resolve. In general, the primary issue with the Artificial Contract UTM is that English law courts would probably refuse to enforce this contract term on the grounds of a lack of certainty, impossibility, or lack of intention to create legal relations[10].

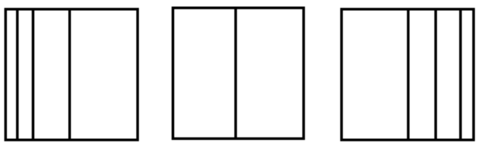
3.2. *Organic Contract-Property UTM*

It is also possible to construct a UTM in contract law without the use of explicit terms incorporating instructions to make a Turing machine. For example, imagine a contract with 19 parties, who we will call parties A, B, C… S. The contract terms state that 18 of these parties are to have property rights in a certain piece of property (α) depending on various conditions.

[10] It is likely the halting problem qualifies as 'conceptual uncertainty': *Re Lehman Brothers International (Europe) (In Administration)* [2012] EWHC 2997 (Ch) at [197]–[198]. Equally, it may qualify as 'impossible' if the halting problem is used to condition some kind of performance. *Forrer v Nash* (1865) 35 Beav. 167, 171. It might also qualify as a contract made 'in jest' or 'as roleplay' *Sutton v Mishcon Reya* [2003] EWHC 3166 (Ch), [2004] Fam. Law 247 at [26].



The ownership rights of the different parties in the object would constitute the symbols. Provided we use moveable property the ownership transfers instantly without the need for registration and thus there is no requirement for human action[11]. To produce cells, we can divide the ownership of the property in space or time, using units of property space or units of consecutive time periods. There are a variety of ways to achieve the 'infinite' storage space required in a Turing machine: one is to divide the units up via an infinite series given there is no formal legal limit on the size of property capable of being owned or the length of time it can be owned for[12]. For example, looking top-down on our cube:



Split parts of α to the left and right. Then, subsequent portions of these divisions can be split into half themselves, producing an infinite series of portions.

[11] *Cochrane v Moore* (1890) 25 QBD 57 (CA). intention alone is sufficient to pass moveable property in a contract for the sale of goods; further, according to Sale of Goods Act 1979 s17, property passes when the parties 'intend it to pass.'

[12] Albeit legal limits may arise tangentially through planning law and interference with aerospace.

To indicate where the head was it would be possible to give the 19th party, party S, a property interest in the portion/cell which is being read. The same could be achieved by taking some definite period of ownership – say 4th July 2022 08:00-09:00 GMT – and dividing up time on either side of 08:30 in successive halves. The series of incrementally smaller slices of time will produce the same effect as the physical divisions. Note, that there is the issue of current time catching up to the defined legal time period: this could be remedied by moving the latter forward relative to the former, with its full register of property rights, to avoid overlapping with the time of calculation. An alternative tape mechanism, to avoid an infinite series and absurdly small periods or physical dimensions of ownership, is to define some starting date, say '3rd of July 2022', and then arrange the tape like so:

Date	…	9th	7th	5th	3rd	4th	6th	8th	…
Tape Cell	…	Left 3	Left 2	Left 1	Central 0	Right 1	Right 2	Right 3	…

Note there is nothing distinct about 'Central 0', other than its function as a point to invert the counting. Say the head was defined by Party S's ownership in the time period. This could be moved 'left' from right 3 (the 8th July) by moving S's ownership two days earlier progressively until the 3rd, at which point S's ownership would increase in length, albeit on odd[13] days, two days at a time to represent the left side of the tape.

The ownership rights of the 18 other contractors would constitute the alphabet of the Turing machine. The two states could be produced by flipping property rights in another piece of property. For instance, if Q owned property β it would be State 1, if they didn't and R owned it, it would be State 2. The contract could then incorporate numerous conditions determining where the property rights were to go in the different cells corresponding to the instructions in Rogozhin's (2, 18) table above. Let us use the non-infinite series model of cell counting. Say R had a property interest in the object on the 5th July 2022 (indicating the head was reading that cell) the contract would state:

[13] Note, 'odd' and 'even' are used here for convenience. However, given the uneven months render this unreliable it will be necessary to define days according to a metric outside the Gregorian calendar dates. One possibility is to use phrasing like 'every other day'; another would be to use time defined according to an atomic clock.

> If Party Q owns Property β, and party S has just been given a property interest in Property α on day *x*, and party A also has a property interest in α on the same day, transfer Party A's interest to Party C, then give party S an interest in property α in two days from the day they previously owned, if even, or two days earlier if odd, (unless the day is the 3^{rd} July 2022, whereby it shall be property on the 4th) and transfer Party Q's ownership of Property β to Party R.

Which corresponds to the line q_2 bb_2 Rq_1 in Rogozhin's table of instructions (where q_2 is state 2, symbol b is the same as Party A having property rights, b_2 is equivalent to Party C having property rights, R is moving right, and q_1 is state 1). Note that the 'read/write' function of the head is covered by the transfer of property rights: A's property interest is wiped off and replaced with Party C's. The full range of conditional statements in the table of instructions can be implemented via a long list of contract terms giving different conditions. It is also possible to implement Rogozhin's halting command by specifying that, in a certain configuration of property rights, the contract will be terminated for all parties. Accordingly, determining generally whether the parties have any contractual rights and duties ahead of time for this contract is equivalent to determining whether the Turing machine embedded in the contract will halt and, as such, is equivalent to the halting problem.

This is merely one possible example of a Legal UTM using the most basic legal rules and institutions. One can imagine all manner of legal rights and obligations, especially in intangible property, which could circumvent the practical problems with using physical property. Nonetheless, the benefit of the UTM above is that none of the contract terms are uncertain or absurd, just unusual, and as such there are no obvious legal issues with the implementation of the contract. Further, unlike the Artificial Contract UTM, one can imagine variations of the commercial arrangement above which are not designed to be Turing-complete but instead accidentally achieve the necessary complexity. Given it would not be obvious the commercial arrangement was a Turing machine, and thus susceptible to the halting problem, these may produce problems later at court.

4. *Implications of the Legal UTM*

If the above is accepted, then English Law is Turing-complete and possesses the characteristics listed in Part 1. These have a range of implications: three will be presented in this paper. First, the philosophical implications of legal Turing completeness are considered. A computing system must have certain properties to function as Turing-complete and these properties differ in their compatibility with two legal philosophies: Dworkinian jurisprudence and legal positivism. A legal UTM is only possible under a positivistic model of law. Second, assuming the law is positivistic, it suggests any computer program can theoretically be run using a legal Turing machine and, more importantly, this legal system can itself be simulated. Broader themes on the possibility of embedding, whether spontaneously or deliberately, systems within other systems are explored. Third, the paper will consider how the properties necessary for Turing completeness have particular social utility. Two theoretical models are used to develop this inquiry: Luhmann's Systems Theory and Weber's account of calculability in Rational Formal Logical (RFL) Legal systems.

4.1. *Jurisprudence and the Legal UTM*

The Turing completeness of law has implications in legal philosophy. In particular, the compatibility of the legal UTM with different conceptions of law. First, we will examine the qualities and characteristics a computing mechanism must have to qualify as a Turing machine. Second, the possibility of computing systems composed of human thought processes, and the requirements those thought processes must possess, are explored. Third, two different models of law, Dworkinian jurisprudence and legal positivism, are outlined and there is a discussion on whether law under either model can meet the requirements of 'Automaticity.'

4.2. *Computing Mechanisms and the 'A-Machine'*

A Turing machine is a hypothetical computer used to demonstrate the nature of computing: as such it exists as an abstract concept. If we want to actually carry out computations, however, we will need some kind of physical device capable of storing data and carrying out operations[14]. There are numerous examples of devices, mechanical or otherwise, which demonstrate Turing completeness[15]. A classic example is a mechanical device with tape and mechanisms which can alter and read the tape. Another is where a human being uses a pen and paper to draw the tape and symbols whilst following the table of instructions. Many systems are

---

[14] PETZOLD C., *The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine*, cit., p. 325.

[15] Charles Babbage and Ada Lovelace's hypothetical 'Analytical Engine' was made of gears, for example.

'surprisingly Turing Complete': for instance, fantasy card games[16], Microsoft PowerPoint[17], and biological matter.[18] Given the diversity of these 'computing substrates', we can ask: what qualities are required in a physical computation machine for it to qualify as a Turing machine? Turing himself gives us a hint: he calls his invention the 'Automatic Machine' (A-Machine) distinguishing from what he calls 'Choice-Machines.' (C-Machine). Choice machines are devices whose operation involve human discretion: *"If at each stage the motion of a machine (in the sense of § 1) is completely determined by the configuration, we shall call the machine an "automatic machine" (or a-machine). For some purposes we might use machines (choice machines or c-machines) whose motion is only partially determined by the configuration (hence the use of the word "possible" in § 1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix a-[19]."*

An important point is that Turing did not believe machines and humans were fundamentally different in terms of their capacity for computation. It is not the mere presence of a human being which determines whether a device is a choice-machine or not – rather, it is role of human choice in the operation of the machine.[20] It is helpful to consider some of the examples above from this perspective. Take the mechanical tape reader: we can convert this into a C-Machine by adding several buttons which alter the output of the machine independently from the table of instructions operated by human beings. Or, following the deeper implications of Turing's definition of the A-Machine, we could consider the same mechanical tape

---

[16] CHURCHILL A., BIDERMANN S., HERRICK A., *Magic: The Gathering is Turing Complete,* in *arxix,* 2019.

[17] WILDENHAIN T., *On the Turing Completeness of MS Powerpoint.* Unpublished manuscript. Access here: Virtually all programming languages are Turing complete. For a good survey, see GWERN., *Surprisingly Turing-Complete,* in *Gwern.Net.,* 2012.

[18] BENENSON Y. *Biomolecular Computing Systems: Principles, Progress, and Potential,* in *Nature Review Genetics*, 2012, p. 455. For an application using DNA BONEH D., DUNWORTH C., LIPTON R., SGALL J., *On the Computational Power of DNA,* in *Discrete Applied Mathematics,* 1996, p. 79. And one using neurons in the human brain: MCCOLLOCH W., PITTS W., *A Logical Calculus of the Ideas Immanent in Nervous Activity,* in *Bulletin of Mathematical Biology,* 1990, p. 99.

[19] TURING A.M., *On Computable Numbers, With an Application to the Entscheidungsproblem*, cit., p. 232.

[20] PETZOLD C., *The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine,* cit., p. 189.

reader with a mechanism linked to a radioactive isotope: when the isotope decays, a random operation is carried out. Neither example is a Turing machine because neither has its outputs 'completely determined' by its prior configuration (following its Table of Instructions). Instead, elements of randomness (human choice, radioactive decay) intervene to disrupt the patterned algorithm.

4.3. *Human Computing Machines*

It is easy to imagine a mechanical system operating automatically according to the laws of physics and its internal mechanism. A more difficult example is a human being following a table of instructions with pen and paper[21]. It is tempting to treat this as a C-Machine because the human being could always refuse to follow the table of instructions and thus disrupt or halt the machine's operation. Does this mean that, contrary to our stock example, human pen and paper computation is not Turing Complete? The answer is no: consider an analogy with the mechanical Turing machine. Every mechanical Turing machine must be constructed and set in motion by a human being, equally, they are vulnerable to disruption by human beings and natural occurrences. Although the machine would carry out its computation automatically according to its mechanism, a human could choose to hit it with a hammer randomly. Nonetheless, the machine is not a Choice Machine: *following its mechanical design, free of external interference* the machine will run automatically according to the table of instructions. Each stage of motion is, in this ideal world, *completely determined* by the previous state of the machine. Turning back to the human with pen and paper, we could clearly point to their disobedience and suggest they had deliberately broken the 'system', in the same way as someone who built a mechanical Turing machine 'broke the system' by smashing it with a hammer. The analogical question then becomes: is the automatic mechanical operation of a device (eg, springs, steam, gears) equivalent to the automatic operation of a table of instructions by a human according its logical rules? It helps to sharpen what we mean by 'automatic mechanical operation.' Let us define it as follows: the machine will make the same movements and produce the same outputs from the same input on any repetition provided the components and the environmental conditions are the same, regardless of who observes the machine or sets it up. In philosophical terms, we could say: the

[21] Turing's own example. TURING A.M., *On Computable Numbers, With an Application to the Entscheidungsproblem*, cit., p. 249.

operation of a machine is automatic, and thus not a ‘choice machine’, if its operation is (A) mind-independent and (B) not random. This is an idealised standard which no real physical machine could ever meet: no components are ever perfectly identical, and no mechanical operation is ever totally free from random malfunction or distortion caused by fluctuating conditions. Nonetheless, we can say that for practical purposes mechanical devices meet these criteria because, whilst not totally free from randomness, mechanical machines are remarkably consistent and when they break down we can usually point to the malfunction.

Can human minds carrying out instructions in a table meet these high standards? What is compelling about Turing’s pen and paper example is that the instructions seem perfectly uncontroversial[22]. They constitute basic pattern recognition and functions which any operator should be able to memorise and follow. Nonetheless, equally they seem to require at least some kind of interpretation – the operator must learn how to recognise the patterns and how to carry out the instructions[23]. This interpretive step casts doubt on the mind-independence of the process: even if different humans interpret and operate the instructions identically, can it be said this is the same as the identical operation of mechanical structures following the laws of physics? As long as we include the necessary interpretative capacities as part of the ‘mechanism’ of the human machine, it is identical. We know of systems of instructions which generate extremely consistent intuitions about correct application amongst all trained interpreters. One example is maths: when you have learned basic arithmetic, you can carry out an infinite number of different sums which, even if never computed before, can be checked for correctness by anyone else who has learned the necessary skills. Further, if you make an error, the correction by someone who spots an error will be accepted by all those who understand arithmetic, and they will do so independently without communication or conferral. Impressively, this appears to hold true across time and space: anyone, in any culture, would accept this correction provided they understood the basic rules of arithmetic and were calculating exclusively according to

[22] These raise difficult questions related to the famous *analytic/synthetic* distinction. Post-Quine, the tautologous or ‘self-evident’ nature of Turing’s operations become dubious; instead, we might regard them as a particularly fixed part of the web of belief, and thus susceptible to re-evaluation following sufficiently strong counterevidence.

[23] Petzold notes this objection – in the fluctuation of the human mind – “*represent a fundamental clash between those who believe the mind to be ultimately a mechanical process of the brain, and those who do not*.” PETZOLD C., *The Annotated Turing: A Guided Tour Through Alan Turing’s Historic Paper on Computability and the Turing Machine,* cit., p. 192.

mathematical standards. Pickering describes a similar phenomenon in the context of science: *"[s]uch disciplines—acquired in training and refined in use—carry human conceptual practice along, as it were, independently of individual wishes and intents. The scientist is, in this sense, passive in disciplined conceptual practice*"[24]

Accordingly, whilst perhaps not fully independent from human beings, the table of instructions in Turing's human pen and paper a-machine can demonstrate the necessary practical characteristics: once explained and given to any person, anywhere, and in any time, the same operations will be carried out according to the same inputs and, if an error is made, corrected equally, universally, and independently[25]. The human being who chooses not to follow the table of instructions becomes like the machine struck with a hammer: the process which would otherwise occur equally and consistently across machines built the same way, or humans instructed according to the same table, was interrupted by a disruption.

The discussion can be developed in two ways. First, an a-machine's operation must be closed. This is an obvious implication of the definition of automaticity: there can be no introduction of inputs outside the machine alphabet and no role for operations outside the basic ones permitted in the Turing machine. For example, a device would not be a Turing machine if one of the operations was 'If Symbol Y in state 3, move forward and continue to do so unless it is raining today, then move left etc.' Instead, the machine's operations are limited to variations of reading symbols in the alphabet, re-writing them, and moving left or right. We can state this requirement in terms of self-reference: the machine's operations (reading and writing) must exclusively reference the symbols in its own alphabet. Any inputs must be first translated into those symbols[26]. Second, the operation of the machine must be unambiguous. For mechanical devices this is reducible to the restriction on randomness: the next computation must not be uncertain because random chance can intervene in the operation of the machine. For human systems this is slightly less obvious and merits a separate requirement. As noted above, automatic human systems require certain properties: that any person instructed in the basic

[24] PICKERING A., *The Mangle of Practice: Time, Agency, and Science*, Chicago, 1995, p. 115.

[25] This move merely side-steps the analytic/synthetic division with a sociological substitute. The universal, independent consensus (measured empirically) of Turing's instructions does not appear to meet the strict requirements for it to qualify as 'analytic.'

[26] For similar discussions in the context of formal systems in general, see HOFSTADTER D., *Gödel, Escher, Bach: an Eternal Golden Braid,* New York, 1979, p. 32 and the 'MU' system.

rules of the system will carry them out the same independently, or accept correction independently. If there is ambiguity when it comes to the next operation such that our human-components are unsure of what to do, or what the right answer is, then it cannot be automatic. Instead, human choice is required, on the part of the individuals choosing the next operation to carry out, or potentially collectively if the community is forced to come together to work out what the 'right answer' ought to be[27].

4.4. *Is Law an A-Machine?*

Having outlined the requirements for an a-Machine in section 2, we can now turn to consider whether law possesses them. As noted above, this question is philosophically controversial. Some accounts of the concept of law are more compatible with the requirements above than others. We can contrast two approaches to reveal these differences: Dworkin's legal philosophy and legal positivism. Turing to the former, there are multiple readings of Dworkins work: the one adopted here is that all legal decisions require the application of moral and political principles[28]. Furthermore, that as a result of his commitment to moral realism, the application of these principles in a concrete case may produce 'one right answer'[29]. The relevance of these two claims, taken each in turn, reveal several intriguing implications for the legal UTM. First, Dworkin's suggestion moral and political considerations are essential to *all* legal decision-making imply that even uncontroversial legal rules, such as the application of the speed limit, raise moral questions. As Kramer notes, Dworkin provides an account of law which treats uncontroversial ('easy') cases as a consensus of moral convictions. He makes a useful analogy with lying: a group of people may all uncontroversial believe lying is wrong, but they would each hold that belief even if they had to hold it alone[30]. The fact of consensus is irrelevant to the individual belief being held. The problem this model presents to the legal UTM is clear: human choice, in the form of reflection

---

[27] This is at the heart of Godel's criticisms of Turing's analogy between human and mechanistic computers.

[28] This is the so-called 'One System' reading of Dworkin, in which law is a *sub-set* of morality. For a recent example, see CRUMMEY C., *One System Integrity and the Legal Domain of Morality,* in *Legal Theory,* 2022, p. 1.

[29] BIX B., *Law, Language, and Legal Determinacy,* Oxford, 1995, p. 77. See DWORKIN R., *Objective and Truth: You'd Better Believe It,* in *Philosophy and Public Affairs,* 1996, p. 87, at p. 119 for Dworkin's model of moral objectivity.

[30] KRAMER M., *In Defence of Legal Positivism: Law Without the Trimmings,* Cambridge, 1999, p. 146.

on what is morally right, is required in every legal case. Even the most cut-and-dry legal rule, such as the application of the speed limit, or conveyance of property following basic formalities, may be legitimately overturned if the judge deciding it comes to a different moral and political evaluation[31]. If law is structured as Dworkin suggests, then the Legal UTM is actually a Legal Choice-Machine and therefore not Turing-complete. Another problem is that the legal system is radically open under this account: any decision can involve the introduction of new, previously unseen moral principles. This is subtly different from the human moral choice issue: the problem would remain even if the principles could be applied consistently and predictably – for instance rigidly following a deterministic utilitarian calculus. Even with such a calculus, no decision ahead of time could be treated as *perfectly determined* by the law because the prevailing stable pattern could always be upset by changing moral consensus; newly developed principles; or advancement in the field, such as economics, altering the application of existing principles[32]. Dworkin's second claim appears to avoid these issues: he suggests that there is a right answer to any legal dispute because there are, in his view, correct answers to any question of political morality[33]. Thus, even though his account appears to allow the judge massive discretion and freedom, in reality the judge has no discretion when it comes to reaching the 'right' decision. This would appear to restore the possibility of the Legal UTM under Dworkin's account: this 'right' decision follows automatically and does not depend on who is reaching the decision. Nonetheless, the debate on moral realism aside, Dworkin's account is still not entirely satisfactory. Whilst it is true that hypothetically a 'One Right Answer' legal system would allow for the construction of a Legal UTM, Dworkin's model does not allow for this in practice. Unless participants can determine what the right answer to the legal question is in advance, independently from one another, then the law does not exhibit the necessary qualities to be 'automatic.' Dworkin's account of Hercules and his derived correct answers provides no account of how any reasoner could determine this right answer and, in fact, seems to suggest the existence of split opinions is intrinsic to the legal process[34].

The above suggests that Dworkin's legal philosophy is incompatible with the Legal UTM. We can now compare legal positivism as an

---

[31] *Ibid.*

[32] See DWORKIN R., *Law's Empire*, Cambridge, 1986.

[33] DWORKIN R., *Objective and Truth: You'd Better Believe It*, cit.

[34] DWORKIN R., *Taking Rights Seriously.* Cambridge, 1977, p. 335–8; BIX B., *Law, Language, and Legal Determinacy,* cit., p. 77.

alternative. The basic thesis of the legal positivist is that law and morality are separate; what is legally valid in a system is not necessarily what is morally acceptable[35]. The positivistic model of law seems to possess many of the characteristics which are required for an a-Machine to function. We will go over each of these in turn. First, many positivists are committed to the position that the complete set of legal outcomes in most legal systems can be more limited and constrained than the total set of outcomes required by morality: *"If law and morality are conceptually distinct, it would be open for a society to recognize as its legal universe some set of grounds for legal decision that is not coextensive with that society's moral universe"*[36].

Schauer describes this as the 'formalism' of law and, as a cultural phenomenon, 'rule-ness'[37]. He argues that one of the distinctive roles of law is to provide an answer which *diverges* from the 'all things considered' optimal decision. Instead, legal outcomes are frequently restricted by the legal sources, sometimes causing them to diverge from what we would do if given complete discretion.[38] Law achieves this restriction through a set of narrower, often *closed* outcomes, set out authoritatively by the legal sources. As a result, legal decisions are not revisable according to the 'optimal outcome' in every case[39]. Second, following from this, positivists typically propose legal reasoning is frequently determinate. They argue that legal rules, whilst not providing answers in all cases, have a 'core' of uncontroversial application[40]. Examples often given are of the speed limit; obeying basic formalities for conveyancing; and procedural law. When it comes to this core, the positivist follows the doctrinal legal scholar and will suggest there is a 'legally correct answer' to the legal problem. Further, they will make the more ambitious claim that anyone familiar with the basic rules of the legal system would reach the same conclusion

---

[35] GARDNER J., *Law as a Leap of Faith: Essays on Law in General.* Oxford, 2012, p. 19

[36] SCHAUER F., WISE V., *Legal Positivism as Legal Information,* in *Cornell Law Review,* 1997. p. 1080, at p. 1088.

[37] SCHAUER F., *Formalism,* in *Yale Law Journal,* 1988, p. 509; SCHAUER F., *Ruleness,* in DUPRET B., COLEMANS J., TRAVERS M., (eds) *Legal Rules in Practice,* Thames, 2022.

[38] *Ibid.*

[39] *Ibid.*

[40] LEITER B., *Legal Indeterminacy,* in *Legal Theory,* 2009, p. 481; LEITER B., COLEMAN J., *Determinacy, Objectivity, and Authority,* in *University of Pennsylvania Law Review,* 1993, p. 549.

independently from one another and, in the event of disagreement, would unilaterally accept the same correction[41].

There is therefore an asymmetry between the legal UTM and different concepts of law. If you find the Turing machine in Part 1 plausible then you are also more likely to endorse a positivistic concept of law. Conversely, if you were suspicious of the possibility of the device, then a Dworkinian or natural-law based concept will be more persuasive.

4.4. *Turing Completeness and Embedded Systems*

Assuming that law, in at least some legal systems, is positivistic, and a legal UTM can therefore exist, what does this mean for law's computability? As mentioned in Part 1, a system with Turing completeness can be used to carry out any kind of computation[42]. Theoretically, a Legal Turing machine can run any program a modern computer can, such as the Windows OS or the Aladdin financial management system. More radically, if of sufficient size, the Legal Turing machine could also be used to run machine learning programs – in fact, the very programs currently being used in attempts to simulate legal systems. These raise interesting questions about the computability of law. This area is hotly debated, with some academics arguing the central purposes and functions of law can be reproduced by a digital system; others arguing that these artificial legal systems will lack something essential[43]. The existence of the Legal UTM is relevant here and a point in favour of the computability of law. If the description number of the Legal UTM can be produced, this could be fed into a digital Turing machine which would then simulate the operation of the legal rules. If one accepts the possibility of constructing a Turing machine within a legal system, one must also accept, therefore, that at least part of the law is capable of being simulated with a computer. There are several strategies to avoid or mitigate this argument. One is to suggest the legal UTM is not Turing-complete in the first place, explored above. Another is to argue that the legal UTM is no longer producing legal outputs – it may operate using legal rules, but its output is not legal[44]. It is helpful

[41] LEITER B., COLEMAN J., *Determinacy, Objectivity, and Authority,* cit., p. 620; a quality described as 'modest objectivity.'

[42] PETZOLD C., *The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine,* cit., p. 143.

[43] For a recent overview, see the essays in DEAKIN S., MARKOU C., (eds) *Is Law Computable?*, London, 2020.

[44] The following is a basic application of the concept of 'code' from systems theory. See LUHMANN N., *Law as a Social System,* Oxford, 2008.

here to draw an analogy with other kinds of embedded legal systems. Consider a standard board game or video-game, say Monopoly or Age of Empires, which lacks law as an explicit game feature. These games have a series of rules which the players must follow, connected to a 'win-condition' determining when players win, tie, or lose. According to the logic of the game, any action is connected to one of these end-states[45]. For example, building hotels in Monopoly is a means to increasing the amount of money other players pay when they land on the builder's property, thus increasing their likelihood of their bankruptcy and therefore the victory of the hotel-builder. Nonetheless, it is hypothetically possible to produce a legal system within these board games which, although using the game resources, converts the outcomes into legal ones[46]. Consider a Monopoly player who uses their position of overwhelming advantage (for instance, large outstanding debts from all the other players) to implement a series of rules which punish players for actions outside of the normal game rules. These rules could prohibit building too many hotels, 'unjustified' rent-collection, or mandatory trades of property to distribute them equally, all enforced by asking for debt collection from the players (thereby threatening their loss and exclusion from the game). Disputes in the application of these rules would be the subject of adjudication in-game by some appointed judge; their decisions may even produce precedents over time which bind future judges. Note, however, that although the resources, mechanism for punishment, and available actions being regulated are all built out of in-game features, the outcome of the dispute resolution is not in-game victory or defeat but typical legal outcomes like 'guilty' and 'not-guilty', or 'legal' and 'illegal.' The logic of the game rules has been converted into the logic of the legal system. The players, though technically in a game of Monopoly, find themselves playing a very different 'game'[47]. We can also consider the reverse – using the legal

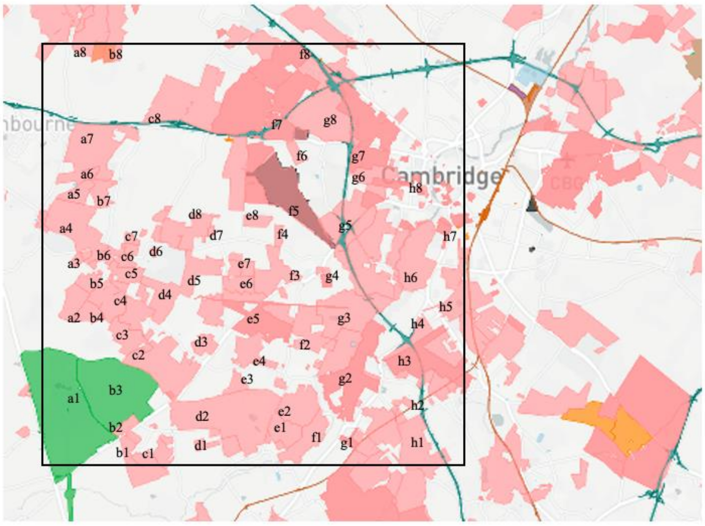
[45] This can produce uncomfortable results. Consider the game 'Scramble for Africa' where you suffer an 'atrocity penalty' for murdering slaves. Critics have accused it of reducing serious historical events to another in-game strategy to win the game. DRAPER K., *Should Board Game Players Play the Role of Racists, Slavers, and Nazis?* In *The New York Times.,* 2019. We can frame this in system terms by suggesting the moral language has been converted into game language.

[46] Or in practice – it is straightforward to create a legal system in Age of Empires once the player/sovereign has amassed significant advantage and is playing with people willing to cooperate with their imposed rules.

[47] This 'code-conversion' is also described as 'juridification', especially in the context of politics where contentious public issues are reframed in the terms of legal debate. See SIEDER R., *The Juridication of Politics,* in FOBLETS M., GOODALE M., SAPIGNOLI M., ZENKER O. (eds) *The Oxford Handbook of Law and Anthropology.* Oxford, 2020.

system to implement a game. One could set up legal rules and actions of litigants in such a way that it mirrors a board-game or a card game. For example, dividing pieces of land using algebraic chess notation to produce a chessboard and then shifting property rights in them to represent moving chess pieces (see *fig 1)*. The legal rules in play here are ultimately connected to outcomes in court, such as who owns the property and how their rights will be protected. Nonetheless, in our legal boardgame the code of the moves has been switched – property is passed as a means to victory or defeat in the chess game rather than victory in court. Legal steps become coded in terms of their victory-relevance rather than their legal-relevance. Transferring a lease to property which is adjacent to the other party's freehold tenancy becomes, in chess terms, moving the Queen adjacent to the other player's King. The litigants find themselves engaged in a distinctly non-legal process.

fig 1.



The same is true of the legal UTM. Although the components of the machine were designed to produce legal outputs in litigation – such as contractual liability, ownership, and enforcement of damages – they are not being used as such. Instead, they have been repurposed to carry out computation. Transferring S's and A's property rights in Property β and Property α would, under normal circumstances, result in different legal outcomes, such as S being able to evict A. They might also result in social evaluations – such as pity, shame, and stigma attached, say, to a bankrupt[48]. In the legal UTM above these outcomes are irrelevant: the relevance of S owning Property β is not their ability to get an eviction order in court but the fact it changes the state of the Turing machine. In the same way that the Legal Chess Game has its own chess game code, the Legal UTM has its own computational code. The fact that the Legal UTM can be converted into a description number and simulated by another Turing machine is not, therefore, a decisive point in favour of the computability of law. At most, it suggests that some of the functions which can be carried out by a legal system can be reproduced by a computer system[49]. This is not a radical suggestion: it has always been recognised, even in the days of Expert

[48] For a good survey of 19th century practice, see FINN M., *The Character of Credit.* Cambridge, 2003.

[49] As Luke Derry pointed out when commenting on this draft, dance steps could be configured to produce a Turing-complete machine – nonetheless, this says very little about the computational qualities of dancing.

Systems, that some functions of the legal system are more susceptible to formal representation by a digital system than others[50]. For example, elaborate tax codes and private law rules[51]. Yet, equally, few conceded the entirety of law could be replaced with digital code. The important question is the extent to which the legal system is composed of formal, computational elements and whether these might have particular social utility.

4.5. *Turing Completeness and Social Technology*

Scholars of modernity typically adopt a positivistic conception of law; interestingly, they suggest the qualities discussed above which enable the construction of the legal UTM are crucial to the functioning of the modern state. This idea has been primarily developed in the work of two German sociologists, Max Weber and Niklas Luhmann. The former sets out the general framework of 'rationalistation' which, importantly, incorporates the notion of 'calculability' closely connected to the legal UTM above[52]. The latter develops calculability, using the ideas of legal algorithms and systemic closure to explain how the law produces certainty[53]. Both connect 'calculability' and the properties of formal systems to the needs of the modern, capitalistic state and historically situate their initial emergence in

---

[50] The core proposal of the expert system is to convert legal knowledge into some kind of formal system which can then be automatically queried. Typical use-cases include basic form filling and generation; tax codes; sale of goods law; and matrimonial property disputes. See STEVENS C., BAROT V., CARTER J., *The Next Generation of Legal Expert Systems – New Dawn or False Hope?* In *SGAI 2010: Research and Development in Intelligent Systems XXVII,* 2010, p. 439; and SUSSKIND R., *The Latent Damage System: A Jurisprudential Analysis,* in *ICAIL*, 1989, p. 23. For arguments about the limitations of Expert Systems MCCLELLAND D., *A Critique of the latent damage Expert System,* in *Information and Communications Technology Law.* 1998, p. 15.

[51] Sometimes referred to as 'computational law.' See MERIGOUX, D., CHATAING N., PROTZENKO J., *Catala: a Programming Language for Law,* in *Proceedings of the ACM on Programming Languages,* 2021, p. 1. It is possible highly bureaucratic areas of law have been 'de-differentiated' and operate according to the relevant administrative logic rather than that of the law.

[52] WEBER M., *Economy and Society (English Edition),* Roth, 1968. TRUBEK M., *Max Weber on Law and the Rise of Capitalism,* in TREVIÑO A., *The Sociology of Law*, London, 2008. TREIBER H., *Reading Max Weber's Sociology of Law,* Oxford, 2020. MELISSARIS E., *Is Common Law Irrational? The Weberian 'England Problem' Revisited*, in *NILQ*, 2004, p. 378. SUGARMAN D., *In the Spirit of Weber: Law, Moderity and 'the Peculiarities of the English'*, in Institute for Legal Studies Working Paper Series 2, 1987, p. 1.

[53] LUHMANN N., *Law as a Social System*, cit.; LUHMANN N., *A Sociological Theory of Law,* London, 1985.

the 18th and 19th centuries. The following section will outline Weber's sociology of law and extend it with Luhmann's autopoetic theory; at the same time, it will link their theories to the a-machine properties discussed above and use this to frame parts of the legal system as a form of calculative-coordinative technology.

The present article is too short for a full excursus of Weber's sociology of law. Nonetheless, one aspect of his theory is of particular interest: the quintessential form of modern law 'Formal, Rational, Logical (FRL) Law.' Weber provided variants of these types, produced via his axes labelled 'Formality' and 'Rationality.' These ideal types correspond to *modes of legal thought* characteristic of societies at different stages of development. The full grid, given by Trubek, appears like so:[54]

TABLE I

THE TYPOLOGY OF LEGAL SYSTEMS CLASSIFIED BY FORMALITY AND RATIONALITY OF DECISIONMAKING PROCESSES

Degree of Differentiation of Legal Norms	Degree of Generality of Legal Norms: HIGH	Degree of Generality of Legal Norms: LOW
HIGH	LOGICALLY FORMAL RATIONALITY	FORMAL IRRATIONALITY
LOW	SUBSTANTIVE RATIONALITY	SUBSTANTIVE IRRATIONALITY

Earlier societies typically have a mixture of informal and irrational law. Legal thought which is informal and irrational involves deciding disputes on a case by case basis according to their particular ethical or emotional merit. Informal and rational legal thought, on the other hand, involves consistent decision-making following general substantive principles, such as utilitarianism, but remains informal because there is little differentiation from other systems of norms. Finally, formal and irrational law captures decision-making processes like the ordeals, which though highly formal (distinguished from religious and moral decision-making generally), have no coherent principle of decision-making[55]. Only the modern society, with its capitalist market and outcome-oriented, efficient government, possesses the most advanced form of law: 'Formal, Rational, Logical' law. FRL law has several important characteristics: it demonstrates high formality

[54] TRUBEK M., *Max Weber on Law and the Rise of Capitalism,* cit., p. 729.

[55] *Ibid.*

because decision-making using it solely references legal materials as defined by the legal system; it also possesses high rationality, flowing from its logical operations: logical deduction using general rules and principles. Socially, FRL law is accompanied by the emergence of legal experts (in Weberian terms 'Carrier Strata'),[56] who are typically trained in universities and form an elite professional caste. FRL law is particularly useful in the context of land, company, and contract law because its formality ensures it is predictable and stable, thus facilitating business planning. As Laura Ford notes: "*Modern market finance capitalism involves long-term, socioeconomic structures of investment and debt (e.g., bonds typically issued with thirty-year [sometimes even fifty-year or hundred-year] terms for repayment). Such structures would be impossible, particularly for large organizations employing capital accounting methods, without some degree of consistency and predictability in law, which for Weber comes with formal rationality*"[57].

More specifically, Weber suggests FRL's predictability is connected to its 'calculability': legal problems, when given to legal experts, can be resolved predictably in advance using the application of standard analytical tools and legal sources[58]. Nonetheless, how the law achieves calculability, and how this is related to law's formalism, is not a theme extensively developed by Weber; instead, we will need to turn to a later sociologist, Niklas Luhmann. One of the central themes of Luhmann's theory, broadly termed 'Systems Theory', is how knowledge systems, like Law, Economics, and Science, consistently produce, reproduce, and differentiate information inside them, and how these processes contribute to society's functioning[59]. According to Luhmann, the purpose of the legal system is to stabilise expectations and it achieves this by using 'conditional programmes.' A conditional programme takes a specific logical form, the 'if-then' form, setting out the legal consequences of different sets of facts. The conditions captured by the antecedent 'if' clause cannot be disproven by subsequent events: whether something is illegal will not change if subsequent events mean, for instance, that the legal rule is no longer efficient or lacks any practical effect[60]. Instead, inside the system of legal thought, the 'if-then' clause is conclusively instantiated at the moment of

[56] TREIBER H., *Reading Max Weber's Sociology of Law,* Oxford, 2020.

[57] FORD L., *Law and the Development of Capitalism,* in HANKE E., SCAFF L., WHIMSTER S. (eds), *The Oxford Handbook of Max Weber,* Oxford, 2019. p. 100.

[58] *Ibid.*

[59] LUHMANN N., *Law as a Social System,* cit.,

[60] *Ibid.*

legal analysis. To sharpen what this means, consider Luhmann's closely connected notion of 'dedifferentiation.' Dedifferentiation is the process by which the law adopts non-legal metrics of decision: "*Where law's institutions (courts, tribunals, statutory drafting, prosecution services), for example, put into effect the policies of the government without giving due consideration to their legal applicability, overtly base their decisions upon the supposed reliability of witnesses who have wealthy connections, delegate responsibility for deciding cases to panels of scientific experts, actively seek to promote people's well-being or protect them from harm or leave matters to be decided by God's will, the legal system will have become dedifferentiated*."[61]

In these situations, which can be roughly styled as decision-making according to the systems of politics; the rules of social status; science; economics; and religion, law is no longer fulfilling its function of coding, *ex ante,* what counts as illegal or legal. Instead, another system is being used, such as 'decide according to political power', or 'God's will', or 'the findings of scientific experts', all of which are fallible depending on ever-changing circumstances[62]. Only with the law's coding mechanisms can stability be produced. The stability of these coding mechanisms is connected to another feature of the legal system: operational closure. Luhmann describes the working of operational closure as follows: "*The basic question is then how operations produce the difference between system and environment and -because this requires recursivity – how operations recognise which operations belong to the system and which do not. We call those systems operatively closed that rely on their own network of operations for the production of their own operations and which, in this sense, reproduce themselves*"[63].

Law's operational closure is achieved through self-referential rules determining what is 'legal information' (law's unique 'Eigenvalue') and what is not[64]. We can give several examples from common law legal systems: statutes which assert the legal authority of other statutes;[65] rules

---

[61] THORNHLL C., KING, M., *Niklas Luhmann's Theory of Politics and Law*, New York, 2003. p. 41.

[62] PATERSON J., *Reflecting on Reflexive Law,* in KING M., THORNHILL C. (eds) *Luhmann on Politics and Law: Critical Appraisals and Applications.* Oxford, 2006, p. 17 notes this would involve 'everything being determined at the level of cognition.'

[63] LUHMANN N., *Law as a Social System,* cit., p. 78

[64] *Ibid, 124*

[65] For instance, the Parliament Act 1911, s.1(1), s2(1)) regulating when a bill can become an act of parliament. For more detailed consideration, see WARD, D., *The Rule of Recognition and Ordinary Law,* forthcoming.

of precedent which are themselves found in previous cases; [66] and accounts of judicial authority found in the decisions of judges[67]. These are all examples of the sources of law providing the rules determining what counts as a valid source of law, in the process legitimising themselves. Operational closure ensures that the determination of a legal point, and attribution of legal meaning, depends exclusively on legal rules: no contingent factors, such as the political success of a certain party, or the results of future experiments, or economic orthodoxy, alter the application of the law. Instead, the only way for 'external information' to be incorporated into the legal system is through its own programmes, specifically, the rules of its sources – for example, in legislation or through incorporation in a precedent-setting case. In turn, legal information (stably produced through law's self-referential systems) can be used in other knowledge systems, such as economics or the media, as the input of their programmes.

Calculability, therefore, depends centrally on law's non-defeasible conditional programmes and its closed, self-referential system of sorting legal from non-legal information[68]. These features closely align with those required for law to operate as an a-machine. The requirement for operational closure matches both Weber's notion of 'formality' and the positivist commitment to the separation of law and other systems of norms. Operational closure ensures that the determination of a legal point depends exclusively on the existing legal rules: if these legal rules are themselves determinate, then no other factors will interrupt their operation. The requirement of determinacy and non-randomness is then met by the 'conditional programme.' The programme of 'if-then' is non-defeasible and depends exclusively on the existing rules and past facts: the operation of the programme is determined entirely by the previous configuration of the system. In this way, it is very similar to the lines in a Turing machine's table of instructions. These two features guarantee that formal legal reasoning is predictable and can run with the kind of regularity necessary to both simulate a Turing machine and plan a modern economy at scale.

---

[66] For example, the rule in *London Street Tramways v Londy County Council* (1898) AC 375, which sets a precedent for rules of binding precedent.

[67] See *Young v Bristol Aeroplane Co Ltd* [1944] KB 718 establishing when the court of appeal is bound by itself; likewise, *Marbury v. Madison,* 5 U.S. (1 Cranch) 137 (1803) which sets out the powers of the US supreme court.

[68] Cf SCHAUER F., WISE V., *Legal Positivism as Legal Information,* cit., p. 1080.

5. *Conclusion*

This article has sought to demonstrate the possibility of building a Turing machine using the rules of English private law. It then connected this possibility to several interesting features of law, specifically: (A) the philosophical concept of law; (B) the possibility of simulating a legal system; and (C) the sociological importance of Turing-complete law. To conclude we may note the limited scope of this article. At most, it shows that one philosophical account of formal legal reasoning (distinguished, perhaps artificially sharply, from other parts of the legal system and process) demonstrates the formal requirements for Turing completeness. It leaves aside to what *extent* this captures the entire operation and function of law; perhaps at most we can say that it seems to be an integral part of private law modules essential for the *ex-ante* planning purposes of economic actors, and that the calculation of legal results for use in these plans (by appropriate legal experts) is perhaps not so different than the calculations of a Turing machine reading off tape and outputting symbols.