The Second Law, Computation, and the Temporal (a)symmetry of Memory

David H. Wolpert

The Santa Fe Institute, 1660 Old Pecos Trail, Suite A, Santa Fe, NM, 87501 (dhw@santafe.edu)

Abstract

A memory is a system for transferring information from one moment in time to another. Photograph-type memories work by exploiting a collapse of state space flow which "initializes" the memory. The Landauer principle tells us that regardless of whether the collapsing proceeds from the past to the future or visa-versa, entropy must be greater on the collapsed side of such a flow. Consequently, the second law means that the collapsing can only proceed from the past to the future. This establishes the logical necessity of the empirical observation that photograph-type memories can only tell us about the past, and not the future. In contrast to such systems, computer-type memories do not require initialization, and therefore need not be directly affected by the second law. Accordingly, such systems can "remember" the future as well as the past. This is true even for irreversible computers.

Many studies have investigated the relations between various aspects of the different arrows of time [3,4,6,7,13,16,17,19]. Most of these studies take it as a given that the psychological arrow of time derives from the thermodynamic arrow of time (i.e., from the second law of thermodynamics). Yet until now no mathematical proof of this connection has been offered. This has allowed some to even go so far as to make the claim that the two arrows of time are not related at all [14].

Without reducing the psychological arrow of time to a mathematically well-defined phenomenon, there is no way to rigorously prove (or disprove) a relation between the psychological arrow of time and the thermodynamic one. Here and in [20], the "mathematically well-defined phenomenon" is taken to be the human ability to remember the past but not the future. In short, it is presumed that there is no aspect of the psychological arrow of time which can not be explained by the asymmetry of human memory.

This paper is an overview of an analysis of memory systems and their relationship with the second law. (The full analysis can be found in [20].) This analysis shows that the asymmetry of human memory is a direct reflection of the asymmetry of the second law. In this way it explains why the future is both the temporal direction into which "information is dissipated" (due to the second law) and the direction into which "information is preserved" (via our ability to remember the past but not the future). The implication of this analysis is that if the second law "went the other way", then we would remember the future, not the past, and the psychological arrow would point towards the past, not the future.

The analysis reviewed in this paper also shows that memory in (abstract) computers need not be directly affected by the second law. Consequently, such memory can infer the future as readily as the past, and possesses no psychological arrow of time. This is in accord with the well-known reversibility of computation [1,2,5,11,12].

1 What Memory Is

For the purposes of this review, a memory system is any physical system whose state at the present time, t_0, can be proven to provide information concerning the state of the world *external to the memory system* at a time $t_1 \neq t_0$. In other words, a memory system is a means of ferrying information from one moment in time to another. Perhaps the simplest example of a memory system is a photograph on a piece of film. The film is the memory system. Its current state at t_0 (i.e., the image on the film) provides "information concerning the state of the world external to the memory system at a time $t_1 \neq t_0$". Although this particular example of a photograph is asymmetric in time ($t_0 > t_1$), in general no a priori temporal asymmetry is assumed: t_1 can either precede t_0 or come after it.

More formally, let S be the space of possible states of a memory system and let W be the state space of the world external to S. We are given some information about the current state of the memory. For simplicity, assume this information is an exact specification of the current value of S: $S(t_0) = s_0$, where t_0 is the present. In general, we

0-8186-3420-0/93 $03.00 © 1993 IEEE

might also be given some extra information, J, which doesn't directly concern $S(t_0)$. A memory system is used by taking the value of s_0 together with J and then inferring something about the state of W at some time $t_1 \neq t_0$. The only inference tools at our disposal are the laws of physics and, if needed, the information theory principle of entropy maximization (MaxEnt - see [8-10,15]). In terms of probability distributions, a memory system is a system designed so that J, the fact that $P_{t_0}(s \in S) = 0$ for $s \neq s_0$, and Hamiltonian dynamics, used together (along with MaxEnt, if need be), result in a non-uniform distribution $P_{t_1}(W)$.

Intuitively, a memory system will work if we can conclude from J and s_0 that there is an interaction between S and W at some time in-between t_0 and t_1 which serves to correlate $P_{t_0}(S)$ and $P_{t_1}(W)$. Colloquially, s_0 is the "memory" of {the state of W at t_1}, and the memory was "stored", in S, during the interaction between S and W. The sharper the constraints imposed by s_0 on $W(t_1)$, the sharper the memory.

Note the implied existence of an observer in all of this. *Someone* is doing the remembering, i.e., someone is observing $P_{t_0}(S)$ and J and thereby inferring something concerning W at t_1. This observer need not be all-knowing. For example, in general the observer will only have access to some of the degrees of freedom of S. Of course, only those degrees of freedom which are observed can be used to remember something concerning W at t_1.

In addition to $P_{t_0}(S)$, we have three distributions involved in memory: $P_{t_0}(W)$, $P_{t_1}(S)$, and $P_{t_1}(W)$. $P_{t_1}(W)$ is what we (i.e., the observer) wish to deduce. Therefore two of the distributions are possible sources of J: $P_{t_0}(W)$ and $P_{t_1}(S)$. As it turns out, both of these J's can help constrain $W(t_1)$. Systems with these two J's are called *c-type* and *p-type* memory systems respectively. An important distinction between the two types of systems is that p-type systems need an "initialization" process, invariably involving many-to-one mappings, to force a peaked $P_{t_1}(S)$. C-type memory systems require no such initialization. On the other hand, as described below, to work well c-type systems implicitly require that W be tractable enough so that it can easily be evolved through time. In contrast, p-type systems make no demands of W.

There exist other kinds of memory systems besides p-type and c-type. Moreover, there are many subtle aspects to p- and c-type systems (see [20]); such systems are not yet fully understood. This paper is intended only to provide a coarse overview of p- and c-type memory systems.

2 C-type Memory Systems

Useful memory systems which exploit information about $P_{t_0}(W)$ are c-type memory systems. These systems work by evolving the joint system $S \times W$ through time from t_0 to t_1. For example, if $P_{t_0}(W)$ specifies exactly the phase space position at t_0 of the outside world, and if $P_{t_0}(S)$ does the same for the memory system, then we can deterministically evolve the joint system through time to determine the exact state of W at time t_1. To agree with the colloquial use of the term "memory", for these types of systems we usually require that the deduced state $W(t_1)$ varies with the value of s_0. (Otherwise, $J = W(t_0)$ fully specifies $W(t_1)$, and the memory system S is superfluous.)

An example of a memory system which exploits $P_{t_0}(W)$ is abstract computer memory in abstract computers. We deal with "abstract" memory and computers so as to keep the conversation general, i.e., so as to avoid issues concerning particular hardware implementations and so as to avoid issues concerning the physical world external to the computer. This means in particular that here, in this example, the dynamical laws with which we're *directly* concerned are those of the computer's programming language, not those of Hamiltonian physics.

For pedagogical reasons, the discussion here largely concerns computer memory as we humans usually think of it, i.e., computer memory of the past. Consider an abstract piece of Random Access "Memory" (RAM) being used in an abstract computer program. For simplicity, label the address of that piece of RAM as 0001. Assume that at time t_0 there is a particular pattern, s_0, at address 0001. We say that "s0 is the state of 0001 at time t_0". In what sense is s_0 a memory of the state the outside world was in at some time $t_1 \neq t_0$?

The "memory system" in this case is the RAM at 0001. The "outside world" is the rest of the abstract computer, including both the entire program code running on the computer, all of the computer's data-storing RAM (outside of that at 0001, of course), the program counter, etc.

Knowledge of the value s_0 alone (!) tells us nothing about the state of this outside world, at any time. This is because we need some reason to doubt the hypothesis that the memory system has never interacted with its external world. (Without such a reason we have no basis for inferring anything concerning that external world from the current state of the memory). However in general no such reason is provided simply by the bit pattern currently stored in the memory. How could a bit pattern, say 0xA312, *by itself*, imply an interaction with the outside world?

Because of this, the isolated RAM 0001 does not constitute a useful memory system. To transfer data from one moment in time to another (i.e., to have the value of s_0 tell us something concerning the state of the world external to 0001 at a time $t \neq t_0$), abstract computer program memories are c-type. In other words, to infer something concerning $P_{t_1}(W)$, they use $P_{t_0}(W)$ in addition to the value

s_0. They rely on the fact that it's possible to deterministically evolve the entire computer system, program, RAM, and all, through time. The memories of such systems work if it's provable from the computer program that s_0, the current contents of 0001, is a reflection of some data which existed in a certain place in the computer in the past. In such a case s_0 is a "memory" of that data from the past.

As an example, consider the following code segment embedded in some larger program:

```
...
SAVE_LOOP = LOOP;
LOOP = 0;
...
```

We say that {the state of SAVE_LOOP after this piece of code has executed} is a memory of {the state LOOP was in just before this code was executed}. The abstract piece of RAM containing SAVE_LOOP is the memory system. Everything else in the abstract computer is the outside world. In particular, the RAM containing LOOP is part of SAVE_LOOP's outside world.

Define t_0 to be the moment this code segment has finished. Then SAVE_LOOP is a "memory of the state of LOOP at a time $t_1 \neq t_0$" in the sense that we can conclude that the value of the memory system containing SAVE_LOOP at time t_0 tells us something about the state of that system's outside world (namely, the value of LOOP) at an earlier time t_1, before the code segment executed. How do we reach this conclusion that before this code was executed LOOP had the value currently found in SAVE_LOOP? We reach it by logically back-tracking the entire computer program in conjunction with the memory containing SAVE_LOOP. In other words, along with SAVE_LOOP we deterministically evolve all of SAVE_LOOP's outside world, including both the coding sections and the data sections of the program. Not only does the inference we make concern the world external to 0001, the procedure by which we make that inference relies on the world external to 0001.

It is important to note some distinctions between the systems discussed above and human memory. An abstract computer's RAM, when used as a c-type memory to give information concerning a previous state of the computer, is not a p-type memory system (see below). On the other hand, human memory appears to be p-type, and at a minimum is certainly not c-type - to remember something concerning the environment outside of our brains, say what we had for breakfast this morning, we humans do not need to deterministically evolve that environment outside of our brains. Another difference between human memory and computer memory is associated with the fact that computers can be run in a completely reversible manner, whereas human brains are time-asymmetric. (We think forward in time, not backward.) In fact, whereas memory of the future is forbidden to humans, computers can remember the future *even if the program running on them is irreversible*; one simply evolves the joint system $S \times W$ forward in time rather than backward in time.

As an example, consider the following code segment:

```
...
Y = Z;
Y = Y + 1;
...
```

Let t_0 be the moment just before this code segment has executed, and t_1 the moment just after it has executed. The state of {the abstract RAM containing Z} at t_0 gives information concerning the state of something external to that {RAM containing Z} at t_1 (namely, the contents of the abstract RAM containing Y at t_1). It does this via exploitation of knowledge of the state of its external world (i.e., the computer program) at t_0. We have a c-type memory system, *of the future*. We have simply "forward-tracked" rather than "back-tracked" the entire abstract computer.

Of course, in a program containing irreversible steps, we might only be able to infer a limited distance into the future. The parts of the RAM we are tracking forward might get overwritten by other RAM, for example, so that our inference no longer varies with the value s_0. However in a similar manner, irreversible steps in a computation might mean we can only infer a limited distance into the *past*. (For example, the parts of the RAM we are tracking might have been overwritten by some other RAM, and therefore their earlier values do not vary with the value s_0.) There are some slight differences between the two scenarios; the main point is that irreversible computation plays no favorites as far as memory is concerned. It can restrict both forward and backward memory.

In general, in a real computer there are two places that p-type memory processes (along with the many-to-one mappings of their initialization and the consequent gain in entropy) might occur *of necessity*: in the mechanism of the input-output devices, or in the memory mechanism of the human operator, via his or her memory of the initial state of the machine. Both of these are interactions with a system external to the computer as a whole. The actions of an abstract CPU operating on an abstract RAM, independently of the outside world, do not necessarily cause a change in entropy.

Of course, regardless of whether or not they interact with the outside world, in their hardware implementation real-world computers often make use of processes which can be viewed as non-c-type memory systems. However the abstract means by which a typical program running on such a computer "remembers" from one moment in time to another is via c-type memory. This distinction, between the characteristics of the actual hardware implementation

on a real-world computer and the theoretical requirements of that implementation, is analogous to the distinction between real-world computers which are all logically irreversible and the completely reversible manner in which one can design theoretical computers (see [1,2,5,11,12]).

The primary drawback of c-type memory systems is that they require W to be small enough and well-ordered enough so that it's feasible both to have $P_{t_0}(W)$ sharply peaked and to evolve the joint system $S \times W$ through time. The importance of this drawback is most apparent when W is the entire physical universe; few naturally occurring memory systems are c-type. Naturally occurring systems which reach a sharp inference about $P_{t_1}(W)$ instead tend to exploit knowledge of $S(t_1)$; they are p-type.

3 P-type Memory Systems

Besides $P_{t_0}(W)$, the other possible source of J is $P_{t_1}(S)$. Useful memory systems which exploit $P_{t_1}(S)$ are p-type memory systems. Because they are privy only to the states of S, p-type memory systems can not work like c-type memory systems by deterministic evolution of the joint system $S \times W$. To see how a p-type memory system can work, imagine that $P_{t_1}(S)$ specifies s_1, the phase space position at t_1 of the memory system, exactly, and that $P_{t_0}(S)$ does the same for the time t_0. (s_1 is called the "initialized" state of the system, and the process of ensuring that $S(t_1) = s_1$ is called the "initialization" of the system, even if $t_1 > t_0$.) For such a situation we can start with $P_{t_1}(S)$ and calculate the state s'_0 which the memory system should be in at time t_0 if there is no interaction between S and W during the interval between t_0 and t_1. If however $s'_0 \neq s_0$, the provided value of $S(t_0)$, then we know that there must be an interaction between S and W sometime between t_0 and t_1. For cases where there is such an interaction, the resultant perturbation from s'_0 to s_0 is strongly dependent on the state of W at t_1. In fact we can explicitly calculate what states of W at time t_1 are consistent with the information that $S(t_0) = s_0$ and $S(t_1) = s_1$. In this way knowledge of $P_{t_1}(S)$ together with $P_{t_0}(S)$ can set constraints on $W(t_1)$ (see [20]).

"Memory" as it occurs in photographs is a p-type memory system. The initialized state of the system is the (blank) state of the film before being exposed. The fact that the image on the film after exposure is not what it would have been if the film had never interacted with photons (i.e., the fact that $s'_0 \neq s_0$) is the "memory". The information conveyed by the memory is what s_0 and s'_0 jointly tell us about $W(t_1)$ (which in this case is something about the photon field of $W(t_1)$).

Another example of a p-type memory is a meteor crater on the moon. The surface of the moon is the memory system, the (relevant part of the) external world consists of the meteor, t_1 is a billion years ago or so, and $S(t_1)$ is the state {the surface of the moon is smooth}. The difference between $S(t_1)$ and the current state of the lunar surface ($S(t_0)$) allows us to infer something concerning $W(t_1)$ (namely that it contained a meteor aimed at the moon). A similar example is a footprint on a beach. The beach is the memory system, the foot is the (relevant part of the) external world, and $S(t_1)$ is the state {surface of the beach is smooth}.

C-type memory systems can acquire J, the additional information they require (beyond that contained in $P_{t_0}(S)$), simply by expanding the scope of what is observed at the present, t_0. (Instead of just looking at the present state of S, it suffices if they also look at the present state of W.) This is not true for p-type memory systems, because the extra information p-type memory systems require exists at a different time, t_1. How to acquire this extra information from the time t_1 when you are (by definition) stuck in the present, t_0? The details of the answer to this question are quite complex, but one rather obvious point can be made: the acquiring of the needed extra information is made extraordinarily easier if a state-space collapsing process operates in S, taking a multitude of states at the time t_2 to a single state at time t_1. (t_2 is shortly before (after) t_1 if t_1 comes before (after) t_0. For example, if $t_1 < t_0$, the collapsing process is a many-to-one mapping starting with a multitude of possible S states before t_1 and ending at t_1, with $S = s_1$.) Given the existence of such a collapsing process, we only have to *directly* infer that the memory system is in one of the multitude of states at t_2, rather than that it is exactly in the single state at t_1.

Given a p-type memory system with such a collapsing process, it is illuminating to invoke the "Landauer principle" of the theory of reversible computation; any state-space collapsing process, whether it goes from the past to the future or visa-versa, must have higher entropy on the collapsed side of the mapping. (This assumption is taken for granted in [1,2,5,11,12], and is rigorously analyzed in [18].) Together with the second law, this assumption tells us that t_2 must be shortly before t_1, not shortly after it. This in turn means that t_1 precedes t_0, and therefore that all p-type memory systems can only be of the past.

In point of fact, it's not simply that the second law restricts p-type memory systems to be of the past. It seems that the second law is in fact *necessary* for these memory systems *to be able* to record the past. In other words, these systems apparently *exploit* the second law. They do this to achieve their initialization.

In the process of initialization one wants a process which can is guaranteed to leave the memory system in a particular state when it completes (for $t_1 < t_0$) or when it starts (for $t_1 > t_0$). In the real world, we only know how to design processes which are guaranteed to end with a sys-

tem in a particular pre-fixed state, not processes which are guaranteed to start with the system in one. The mechanism we invariably use to achieve our ending-state guarantee is the second law. For example, consider the initialization of the moon's surface to be smooth. This initialization is achieved by thermally relaxing the surface to its highest entropy state. It's achieved via the second law.

Even when the initialization isn't achieved by thermal relaxation of the system by itself, the second law is invariably exploited. For example, beaches are initialized to be smooth by coupling them to external systems (the ocean, the atmosphere), and then relaxing the *combined* system. At the end of this process, the memory system is in its prechosen initialized state. In general, this state might not itself have high entropy. But we arrived at that state by using the second law, to increase the entropy of the joint system consisting of the memory and its external initializer.

In their temporal asymmetry, p-type memory systems contrast markedly with c-type memory systems, which don't require state-space collapsing processes. This asymmetry of p-type memory systems is caused directly by the second law - without this law, p-type systems could remember the future with the same ease (or difficulty) as the past. Under the working assumption that human memory is p-type, this constitutes (the outline of) a proof of why the psychological and thermodynamic arrows of time are correlated.

4 Concluding Remarks

The work reviewed in this paper can be viewed as an extension of the work of Jaynes and many others [8-10,15] on applying the ideas of probability theory to physical systems. Alternatively, this paper can be viewed as related to the work of Bennett and others [1,2,5,11,12] investigating Maxwell's Demon and the thermodynamic behavior of computing devices. The main differences between the work reviewed here and that of Bennett and others are 1) the work reviewed in this paper is more formal in its treatment of memory and has a correspondingly wider scope than the preceding work, extending the concept of memory beyond computing devices, 2) the work reviewed here shows that running an irreversible computer program doesn't necessitate an asymmetric computer program memory, 3) this work shows explicitly that computer program memory and human memory are of a fundamentally different nature, and 4) this work investigates the implications of the analysis of p-type memory systems for the psychological arrow of time.

Acknowledgments

This article was prepared at the Santa Fe Institute, and was partially funded by NLM grant F37 LM00011.

References

[1] Bennett, C. H. (1982). International Journal of Theoretical Physics, **21**, 905.

[2] Bennett, C. H. (1988). IBM Journal of Research and Development, **19** (no. 1), 16-24.

[3] Bitbol, M. (1988). Philosophy of Science, **55**, 349-375.

[4] Davies, P.C.W. (1974). The Physics of Time Asymmetry, UCAL Press.

[5]Fredkin, E., and Toffoli, T. (1982). International Journal of Theoretical Physics, **21**, 219-.

[6] Gold, T. (Ed.) (1967). The Nature of Time, Cornell University Press, Ithaca, N.Y

[7] Hawking, S.W. (1988). Cambridge University report, 2/88 (no report number).

[8] Jaynes, E. T. (1957a). Physical Review, **106**, 620-630.

[9] Jaynes, E. T. (1957b). Physical Review, **108**, 171-190.

[10] Jaynes, E. T. (1982). Proceedings of the IEEE, **70**, 939-952.

[11] Landauer, R. (1961). IBM Journal of Research and Development, **5**, 183.

[12] Landauer, R. (1985). Annals of New York Academy of Sciences, **426**, 2.1.

[13] Layzer, D. (1976). The Astrophysical Journal, **206**, 559-569.

[14] Popper, K. (1965). Nature, **207**, 233-234.

[15] Skilling, J. (Ed.) (1989). Maximum Entropy and Bayesian Methods. Kluwer Academic Publishers.

[16] Wheeler, J.A., and Zurek, W.H. (1983). Quantum theory and measurement. Princeton University Press, Princeton, NJ. 785-786.

[17] Wolpert, D. H. (1988). Nature, **335**, 595.

[18] Wolpert, D. H. (1990). The Relationship Between Many-to-one Mappings And Thermodynamic Irreversibility. Los Alamos National Laboratory report LA-UR-90-4108.

[19] Wolpert, D. H. (1992). *PHYSICS TODAY*, March, 98-99.

[20] Wolpert, D. H. (1992). Memory Systems, Computation, and the Second Law of Thermodynamics. International Journal of Theoretical Physics. **31**, 743-785.