Ant-Based Computing[1]

Loizos Michael*
Harvard University

Abstract A biologically and physically plausible model for ants and pheromones is proposed. It is argued that the mechanisms described in this model are sufficiently powerful to reproduce the necessary components of universal computation. The claim is supported by illustrating the feasibility of designing arbitrary logic circuits, showing that the interactions of ants and pheromones lead to the expected behavior, and presenting computer simulation results to verify the circuits' working. The conclusions of this study can be taken as evidence that coherent deterministic and centralized computation can emerge from the collective behavior of simple distributed Markovian processes such as those followed by biological ants, but also, more generally, by artificial agents with limited computational and communication abilities.

Keywords
Ants, pheromones, modeling, logic circuits, universal computation, collective behavior

1 Introduction

Entomological studies suggest that ants interact with each other and their environment in very specific, usually reactive ways, without individually performing complicated computations, and always in a probabilistic and distributed manner. Such behavior seems to be necessary, and perhaps optimal, for guaranteeing the survival of ants in the uncertain natural environment. In this work we ask what else, besides "mere" survival, such a behavior is appropriate for, what the computational power of antlike protocols is, and how the nature of problems that are solvable by such protocols can be characterized.

Viewed as abstract processes, ants distributively execute simple memoryless probabilistic algorithms. Each process is Markovian, and evolves as a function of the current state of the world only. Communication between processes is very limited, and comes only indirectly through the interaction of these processes with their environment. In this work we propose a model that captures these characteristics in the behavior of ants and antlike processes. We then argue that when coupled with appropriately defined initial conditions, a collective of such antlike processes can reproduce the necessary components of universal computation. We establish this result by showing how the behavior exhibited by these processes is sufficiently powerful to simulate the computation of logic circuits, such as those found in modern digital computers. We proceed to discuss what the initial conditions look like, and how they relate to the design principles of logic gates. Computer simulation results are presented as evidence that the undertaking of a circuit computation by an ant collective is feasible. This feasibility we find rather intriguing, given that the very nature of antlike processes is in direct contrast to the design principles followed by digital computer hardware, namely, binary logic, deterministic computation, heavy reliance on memory, and centralized and complex decisions.

* School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, U.S.A. E-mail: loizos@eecs.harvard.edu

1 A preliminary version of this work appeared as: Michael, L. (2005). Ant-based computing. In M. S. Capcarrère, A. A. Freitas, P. J. Bentley, C. G. Johnson, & J. Timmis (Eds.), *Proceedings of the Eighth European Conference on Artificial Life* (pp. 572–583). Berlin: Springer-Verlag.

The results presented in this work suggest that ants and antlike processes can collectively compute much more than the sum of what their individual parts can: They can perform arbitrary deterministic computations, and can store and retrieve states. In some sense, the Markovian behavior of the individual processes is surpassed by the collective workings of a number of processes. In the collective setting the processes are no longer the computational units, but are simply carriers of information. The carriers exhibit limited intelligence, and the actual computation emerges as a result of the interaction of these carriers through their environment. This same behavior, we believe, may be observed in the study of other social biological organisms, including humans in settings where the social component of their interaction is of primary importance.

Beyond its biological implications, an equally important implication of our study is on the computational power of a multi-agent system. Our results suggest that even if agents in a multi-agent system are endowed with a simple local probabilistic computational mechanism and limited communication abilities, it remains possible for the system as a whole to exhibit arbitrarily complex behavior. This result is relevant to research areas dealing with the study of emergent coherent behavior from unreliable components (e.g., amorphous computing [1]). Other relevant research areas are those dealing with the design of software mobile agents that roam a network and collect pieces of information. Bandwidth considerations impose constraints on the capabilities of such agents, and their interaction could prove to be key in achieving their common goal.

1.1 Related Work

Our endeavor is not the first approach to understanding what the computational capability of ants is. Previous work has shown that ants are capable of solving nontrivial, and in fact presumably hard, problems, such as finding the shortest paths between two points [3], and sorting multiple items into piles [4]. Our goal, however, differs significantly, in that we attempt to investigate whether an ant collective is capable of universal computation, a problem that ants do not, to the best of our knowledge, undertake in their natural environment. This is unlike the problems of finding a shortest path to a food source or the nest, and of sorting the brood, the food, and the dead ants into different piles, tasks that seem to be essential for the survival of ants.

Related fields in computer science that have enjoyed a lot of attention over the past decade are those of ant-based clustering [7] and ant colony optimization [5]. Algorithms inspired by the behavior of ants are employed to heuristically cluster data according to some similarity measure, or to heuristically solve combinatorial optimization problems that are currently thought to be intractable. Despite being inspired by ant behavior, algorithms in these fields often employ state-based ant agents whose behavior is tailored to the actual problem being solved. In addition, ant agents perform only part of the actual computation, with a centralized entity actually monitoring and affecting their behavior over time. In contrast, we are concerned with developing a biologically plausible ant-based model of computation, where ants are memoryless and unaware of the problem under consideration. The solution to the problem emerges only from the collective behavior of ants, and the computation is distributed, without its dynamics being centrally controlled.

Constructing circuits using biological or physical substrates has also been explored in the past. Cells have been manipulated to compute simple logic circuits [6], and fluids have been shown capable of computing the basic logic gate operations [8]. The problem we investigate here complements the existing approaches, in that it provides yet another biologically inspired domain for implementing circuits. We believe that ants are, in fact, a more appropriate metaphor for current flowing through electronic circuits than the different kinds of proteins used in cell-based computing and the different colors of fluids used in fluid-based computing, since unlike the latter approaches, our approach treats all "electrons" equivalently. This makes our approach modular and circumvents the limitations of other non-modular approaches, which can, at the time of this writing, scale up only to the design and implementation of single gates or very small circuits.

On the other hand, it could be argued that unlike the other two approaches for which real-world applications could be presumably given, it is unlikely that the implementation of an ant-based computer

will be of practical importance. We acknowledge that ant-based computers will probably not appear in households any time soon, except perhaps as a novelty item. In fact, although we do not a priori dismiss the possibility, it is beyond the scope of this work to claim that any particular species of real ants would behave as needed to give rise to universal computation. Our goal here is to establish the principled computational power of ant colonies, and show that universal computation could emerge on the grounds that the limited and unreliable decision making of individual ants does not already render this eventuality unattainable. The validity of this result does not hinge on the actual implementation of an ant-based computer; neither does the more practical implication that our study has on the computational power of multi-agent systems, as discussed earlier.

1.2 Overview

In the rest of the article, we provide an answer to the question of whether an ant collective can compute arbitrary logic circuits. We start in Section 2 by presenting the model we employ for ants and pheromones, the two key building blocks of this work. We then discuss the biological and physical plausibility of our model with respect to the behavior of actual ants and the properties of actual pheromones. In Section 3 we describe how one can build an inverter, a basic component of any logic circuit. The engineering side of the problem is first analyzed, and some theoretical analysis and experimental results are then provided that show that the constructed inverter is indeed a functional one. In Section 4 we present a list of additional components needed for building circuits. We go on to show how primitive components can be put together to build the basic logic gates, and how those can be put together to build the circuits one needs when building a computer. Various issues related to this work and the intriguing parallels between ant-based circuits and electronic circuits are discussed in Section 5. We conclude in Section 6 with some pointers to future directions for this work.

2 Modeling Ants and Pheromones

The two key building blocks of an ant-based computer are ants and pheromones. In this section we propose a model of their behavior and properties, and state the assumptions we make for the rest of the article.

Ants are viewed abstractly as processes that operate on their environment on a discrete time basis; their behavior is formally described in Algorithm 1:

ALGORITHM 1: *The Markovian decision-making process of each individual ant at each time step.*

CHOOSEACTION (current location L_C, current direction D_C)

1: Identify the set $\mathcal{R}(L_C, D_C)$ of all locations reachable in one step.

2: Sense the pheromone concentration $\mathcal{P}(L)$ for each $L \in \{L_C\} \cup \mathcal{R}(L_C, D_C)$.

3: If $\mathcal{P}(L_C) \geq T + \varepsilon$ and there exists $L \in \mathcal{R}(L_C, D_C)$ s.t. $\mathcal{P}(L) \leq T - \varepsilon$, then secrete a fixed amount of pheromone at L_C.

4: Choose $L_N \in \mathcal{R}(L_C, D_C)$ with probability $\mathcal{P}(L_N)^n / \sum_{L \in \mathcal{R}(L_C, D_C)} \mathcal{P}(L)^n$.

5: Move to location L_N with direction D_N defined by vector $\overline{L_C L_N}$.

At each time step, each ant is present at some location, has a given direction, and has a set of *adjacent* locations determined by the physical constraints that its environment may impose. An ant's current location L_C and current direction D_C determine, at step 1, which of the ant's adjacent locations constitute the set $\mathcal{R}(L_C, D_C)$ of locations *reachable* within a single time step. The ant's interaction with its environment consists of the ant sensing, at step 2, the pheromone concentrations at its current and reachable locations; choosing, at step 3, whether to secrete pheromone at its current

location; and choosing, at step 4, one of the reachable locations L_N. This chosen location L_N becomes, at step 5, the ant's next location, and determines, the ant's next direction $D_N \triangleq \overline{L_C L_N}$.

The computation carried out by any single ant is, in its nature, reactive. An ant does not keep track of its past sensory input, and can thus base its decisions only on the current state of its environment. The decision of where to move next is taken probabilistically, in the sense that an ant reaching a given state twice will not necessarily make the same choices; nonetheless, the probability distribution over the possible choices in a given state is fixed across the entire history of an ant. In addition, the probability distribution is identical across all ants, although the probabilistic decisions taken by ants based on this probability distribution are independent across ants, giving the ant collective as a whole a distributed behavior. The form of the probability function[2]

$$\Pr(L_N) \triangleq \frac{\mathcal{P}(L_N)^n}{\sum_{L \in \mathcal{R}(L_C, D_C)} \mathcal{P}(L)^n},$$

based on which an ant chooses a location L_N as its next location among the set $\mathcal{R}(L_C, D_C)$ of reachable locations, is consistent with experimental evidence on the exploratory pattern of real ants [3]. The parameter n determines the degree of nonlinearity in the probabilistic behavior of ants, with higher values of n leading to a more drastic behavioral change, given a fixed change in the distribution of pheromone concentrations.

We employ only a single type of recruiting pheromone in this work. Ants are attracted to the pheromone, and in particular are more likely to move toward a location that has a higher than one that has a lower concentration of the pheromone, as described above. Ants are also able to determine whether a sensed pheromone concentration is sufficiently above or sufficiently below some threshold T that is fixed across time and ants. Experimental work with real ants shows that such time-invariant thresholds exist in ants, and that ants adjust their behavior according to whether these thresholds are exceeded by some stimulus [2]. The parameter ε employed in the algorithm accounts for the ants' imperfect sensing, which can reliably detect only whether the pheromone concentration is sufficiently far from the threshold T. In our proposed model, an ant that senses a sufficiently high pheromone concentration at its current location, and a sufficiently low pheromone concentration at some reachable location, will secrete a fixed amount of pheromone at its current location. Due to diffusion, the secretion of pheromone at some location indirectly increases the pheromone concentration at all the adjacent locations. The increased pheromone concentrations help, in turn, to recruit other ants.

In addition to being secreted by ants, a certain amount of pheromone is pumped into the system at specific locations, after the lapse of every time unit. Each pump releases pheromone at a constant rate, although the rate may differ across pumps. A constant fraction of the pheromone at each location dissipates into the environment at every time step. What is more, pheromone diffuses across adjacent locations by flowing from higher concentrations toward lower concentrations, moving closer to a uniform distribution across the system. The change of pheromone concentrations over time is formally described by the equation

$$\mathcal{P}^t(L) = (1-d) \cdot \left[(1-f) \cdot \mathcal{P}^{t-1}(L) + f \cdot \mathcal{W}^{t-1}(L)\right] + s^{t-1}(L) + p(L).$$

Time-dependent functions are superscripted with the time step. The function $\mathcal{P}^t(\cdot)$ maps a location to the amount of pheromone present at that location, while the function $\mathcal{W}^t(\cdot)$ maps a location to the average pheromone concentration of that location and all its adjacent locations. The function

2 In case all pheromone concentrations are zero, the new location is chosen uniformly at random among the reachable locations.

$s^t(\cdot)$ maps a location to the amount of pheromone secreted by ants at that location according to the protocol we have described above. The function $p(\cdot)$ maps a location to the (time-independent) amount of pheromone pumped in at that location during any given time step. The constant d determines the percentage of the concentration of the pheromone that dissipates into the environment, while the constant f determines how uniformly the pheromone will diffuse during any given time step. Initially, all locations have a zero pheromone concentration.

2.1 Biological and Physical Plausibility

The proposed model for ants and pheromones is a plausible one. Our model assumptions on (i) the attraction of ants toward higher concentrations of recruiting pheromone, (ii) the ability of ants to sense pheromones and change their behavior accordingly, and (iii) the secretion of additional pheromone to attract more ants, are based on empirical evidence on the behavior of real ants. Regarding the properties of pheromones, our model relies on two physical principles: (i) gas dissipates into the environment at a rate that is proportional to its concentration, and (ii) gas concentrations tend to become locally (and eventually globally) uniform with the lapse of time at a rate that is proportional to the local gradient of the current concentrations.

One may observe that the assumptions discussed above are in some sense necessary, or minimal, in order to achieve the desired goal of building an ant-based computer. The bias that ants exhibit toward higher pheromone concentrations is essential in that it provides a minimal requirement for an otherwise probabilistic process to behave in some predictable manner. We extract some sort of deterministic behavior from ants by recognizing that the statement that "ants will more likely follow higher pheromone concentrations" is a statement that is always (i.e., deterministically and not probabilistically) true, albeit it, in itself, refers to a probabilistic process. The ability of ants to sense and secrete pheromone is also essential, since it corresponds to a way for ants to read from and write to a shared memory, and thus communicate. The diffusion of pheromone ensures that messages propagate between ants, whereas the dissipation of pheromone ensures that the system purges previously sent messages. Note that despite the minimalistic nature of the communication, some sort of messaging is nonetheless important in order to go from the distributed and independent behavior of individual ants to the coherent and globalized computation needed for an ant-based computer.

Overall, our proposed model and assumptions can be viewed in a more general context and abstracted away from ants and pheromones. The same principles apply when one considers probabilistic processes that exchange simple messages. The environment in which these processes reside is such that it constrains the processes from taking arbitrary actions, and it rewards them for following certain rules, but does not force them to do so. One may then expect that, given sufficiently high rewards, the processes will choose to follow these rules. In some sense one could say that ants move toward higher pheromone concentrations because they find it rewarding. Consequently, our model and results later on are applicable in a much more general setting, and illustrate that robust global behavior can be obtained from distributed probabilistic processes.

3 Building an Inverter

The single most important component in a logic circuit is the inverter; see Figure 1. An *inverter* takes a single truth value as its input and maps it to its negation as its output. In this section we propose a construction of an inverter based on ants and pheromones, and the model thereof developed in the previous section.

3.1 Engineering Analysis

We start with a discussion of how an inverter may be physically realized. We think of an inverter as comprising a set of locations, each represented by a hexagonal cell in Figure 1. Dark cells represent walls that cannot be crossed by ants, and through which the pheromone does not diffuse. Gray cells represent the paths that ants can follow and through which the pheromone diffuses; the assumption

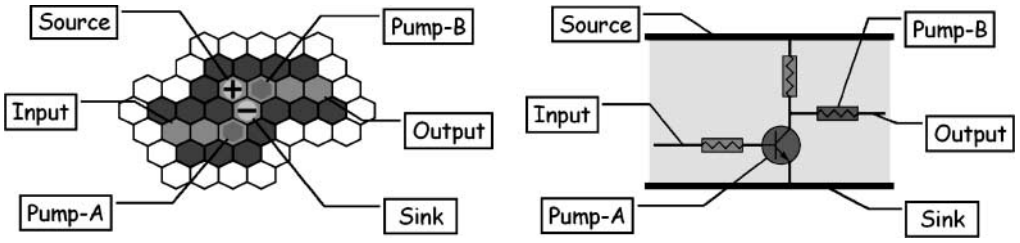

Figure 1. An ant-based inverter and an electronic RTL inverter.

is that these paths are narrow enough so that ants cannot change their direction, unless they encounter a choice point (see, e.g., [3]). There are certain special types of gray cells. Gray cells marked with a plus sign represent sources of ants, where a new ant appears at every time step, while gray cells marked with a minus sign represent sinks of ants; an ant that reaches those cells disappears. Gray cells marked with a darker hexagon represent pheromone pumps; the rate at which pheromone is pumped into the system is implementation-dependent and may differ across pumps. Finally, white cells represent the surrounding area of the inverter.

Ants move within the inverter following one of the paths that are available, depending on the ants' location and direction. Ants enter the inverter at point *Input* and move toward point *Pump-A*, and then toward point *Sink*, where they leave the system. New ants enter the system and the inverter at point *Source*, where they are faced with a choice point. Ants either choose to move toward point *Sink*, where they leave the system, or choose to move toward point *Pump-B*, and then toward point *Output*, where they exit the inverter.

We note here that although the output of the inverter is driven by the input, the ants reaching the output do not come from the input, but rather from the source. The implications of this design and implementation choice are twofold: First, any particular ant traverses only a small distance, since it leaves the system once it enters the next inverter and reaches the inverter's sink. Second, even if an ant starts moving backward in a path, it will eventually leave the system through some preceding inverter's sink. Together, these implications support the scalability of the proposed implementation to large circuits, since ants, both well-functioning and ill-functioning ones, traverse only a distance that does not increase with the complexity of the circuit.

It is worthwhile pointing out the extreme resemblance of the various parts of an ant-based inverter to the parts of an electronic resistor-transistor logic (RTL) inverter, as illustrated in Figure 1. In the former (latter) inverter the input reaches the sink (ground). When ants are (current is) present at the input, the input attracts (drives) the source ants (current) toward the sink (ground), by increasing (decreasing) the pheromone (resistance) compared to the other choice of moving toward the output. Deeper connections between ant-based and electronic circuits are discussed in Section 5.

3.2 Theoretical Analysis

Any implementation of an inverter should respect a necessary set of design principles that make the inverter functional. By definition, the output of an inverter should depend inversely on its input. Equally important, however, are certain requirements that relate to the context in which an inverter is used. The inverter should exhibit sufficient indifference to its input to be able to be driven by the merged output of at least two other inverters, and should exhibit sufficient output gain to be able to drive at least two other inverters.

We define what constitutes a logic 1 and what constitutes a logic 0 for an ant-based inverter by taking the average flow of ants through a given location over a fixed period of time. Since at most one ant may be at each location at each time step, the average flow takes a value in the interval [0, 1]. The inverting behavior then corresponds to requiring the following: When the input flow is sufficiently close to 0, the output flow should be sufficiently close to 1, and when the input flow is sufficiently close to 1, the output flow should be sufficiently close to 0. We therefore define a *logic* 0 to correspond to a flow in the interval $[0, x]$, and define a *logic* 1 to correspond to a flow in the

interval $[y, 1]$, for some choice of the values x and y. Of course, these two intervals should not overlap, and in fact they should be sufficiently apart; we thus require that $x \ll y$.

In order to allow for sufficient input indifference and output gain, it suffices for the output of an inverter to produce amplified signals with respect to the ones defined above. We define an *amplified logic* `0` to correspond to a flow in the interval $[0, x/2]$, and define an *amplified logic* `1` to correspond to a flow in the interval $[2y, 1]$. It is then easy to see that merging or splitting two amplified signals results in a signal that can still be properly recognized by an inverter as a logic `0` or a logic `1`.

In the terminology used in the design of electronic circuits, the intervals $[0, x]$ and $[y, 1]$ described above are known as the *noise regions*, or the *noise immunity levels*, of an inverter. They correspond to the intervals within which the signal is allowed to fluctuate due to external noise, without affecting the inverter's logic state. In the case of ant-based inverters, this "noise" corresponds to the variation of ant behavior with respect to their expected behavior, but it also encompasses the effects of splitting and merging ant flows in a circuit.

3.3 Experimental Analysis

To experimentally validate the proper functionality of the designed ant-based inverter, we have simulated a single inverter and analyzed its behavior in response to different input values. We have recorded, for each time step, the complete state of the inverter, which comprises the presence or absence of an ant, and the concentration of pheromone at each location. We have evolved the state according to the model described in Section 2, by empirically setting the model parameters to appropriate values, as shown in Table 1.

We note that in the simulation we have relaxed the assumption that ants appear at the inverter's source at each time step, to the assumption that this event happens with high probability. The results show that this relaxation does not affect the proper working of the inverter, and indicate that relaxing other assumptions (e.g., that all ants move at every time step) is possible, without unwanted consequences.

The three graphs in Figure 2 show that as the input ant flow increases, the pheromone concentration at point *Sink* also increases, and exceeds, for a sufficiently large input ant flow, the pheromone concentration at point *Pump-B*. This change in the balance of pheromone concentrations increases, in turn, the probability that ants entering an inverter through the inverter's source will move toward

Table 1. Parameters and associated values used in the experimental setting.

Description of model parameters	Empirical values
Probability of new ant appearing at source	95% per time unit
Exponent n in ant probability function	30
Threshold T for pheromone concentration	6 units
Threshold accuracy error ε	0.1 units
Pheromone amount secreted by ants	12 units per secretion
Pheromone dissipation rate d	10% per time unit
Pheromone diffusion rate f	10% per time unit
Pheromone pumped in at point *Pump-A*	1 unit per time unit
Pheromone pumped in at point *Pump-B*	0.2 units per time unit

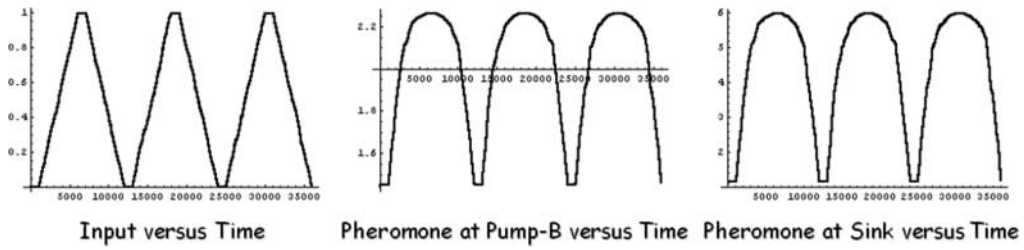

Figure 2. The pheromone concentrations at the choice point of an inverter.

the inverter's sink, which then results in a reduced ant flow at the inverter's output. The output ant flow as a function of the input ant flow in an inverter is shown in Figure 3, where the graphs clearly depict that the two ant flows are inversely related.

Note, in particular, that the third graph in Figure 3 illustrates the amplification performed by the inverter. The width of the line corresponds to the variation in the behavior of the inverter with respect to its expected behavior. Even by taking the least amplified output value at every input value (i.e., the highest output value when considering a logic 1 input, and the lowest output value when considering a logic 0 input), one can see that setting $x = 0.05$ and $y = 0.22$ leads to noise regions that more than satisfy the conditions set forth in Section 3.2. In particular, when the input lies in the interval $[0, x]$, the output lies in the interval $[3y, 1]$, and when the input lies in the interval $[y, 1]$, the output lies in the interval $[0, x/5]$; again, these regions take into account the worst observed behavior in the simulated experiments. The fact that these intervals exhibit an output gain factor of 3 and an input indifference factor of 5 shows that the implementation of the ant-based inverter has noise robustness that exceeds the theoretical requirement for these factors by at least 2.

4 Gates, Circuits, Computers

Beyond an inverter, certain other primitive components are required for implementing logic circuits. We present a list of such primitive components, and describe how each of these can be implemented in a manner consistent with the implementation of the ant-based inverter. We then discuss how the primitive components can be combined into ant-based gates, ant-based circuits, and, eventually, full-fledged ant-based computers.

4.1 Primitive Components

Figure 4 shows the complete list of the required primitive components for building an ant-based computer. We discuss the functionality and implementation of each primitive component next.

A *wire* is simply a path surrounded by walls that guide ants in a given direction. As in the case of the inverter, the paths formed by the walls are sufficiently narrow to prevent ants from changing their direction. The positive battery terminal corresponds to an ant *source*, and the negative battery terminal

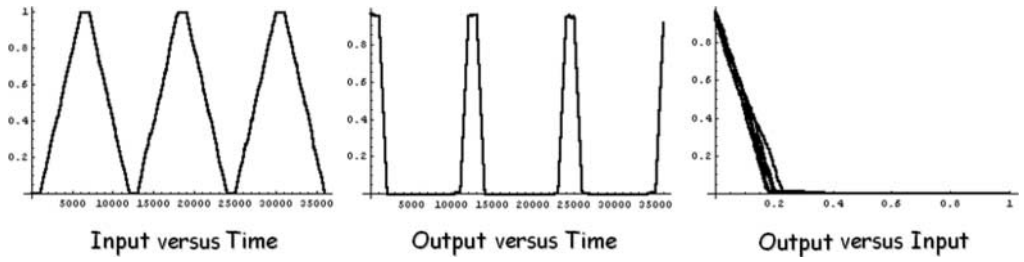

Figure 3. The ant flows at the input and the output of an inverter.

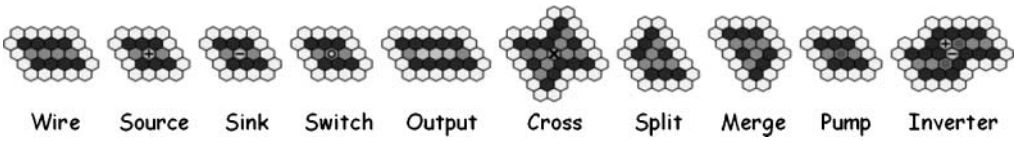

Figure 4. List of circuit components.

to an ant *sink*. A *switch* is simply a controlled ant source where some sort of valve is used to start or stop the flow of ants that enter the system. The *output* of a circuit is simply a designated part of some wire that one can observe; the rest of the circuit need not be directly observable. A *wire-cross* is a bridge that allows one path to pass on top of another, without the two paths ever meeting.[3] A *wire-split* is a choice point for ants, where a path splits into two; the only other choice point is the one in the inverter. A *wire-merge* is a point where two paths merge into one. By appropriately choosing the width of the paths at the merging point, one can physically restrict incoming ants to choose only the outgoing path. Finally, a *pump* is a device that releases pheromone into the system at some prescribed constant rate.

4.2 Basic Logic Gates

One can combine primitive components to build the basic logic gates, as illustrated in Figure 5. It is well known that the set of operators $\{\neg, \vee\}$ is sufficient to express any binary operator. In the case of ant-based computing, the inverter plays the role of the negation operator, while the wire-merge, also known as a wire-or in electronic circuit design, plays the role of the disjunction operator. In order to normalize merged flows of ants, wire-merges are always followed by an inverter or by an `ID` gate. By appropriately negating the inputs and output of a wire-merge, one may easily obtain all of the basic logic gates. Since the ant-based inverter is amplifying its output, it follows that all the ant-based gates are, by construction, amplifying their output.

4.3 Circuits and Computers

The constructed gates and primitive components offer the starting point for one to build circuits with specific functionality. Figure 6 shows the implementation of some basic circuits that can be found in a computer's random access memory and its arithmetic-and-logic unit, where data are respectively stored and processed.

Continuing in this manner, one can keep building more and more elaborate circuits, up to a full-fledged ant-based computer. We do not, however, pursue the design of more complex circuits in this work, since it does not offer any additional insight into our primary goal of establishing the feasibility of such a task. We feel that the circuits that have been presented suffice to substantiate the claim that the task is indeed feasible.

4.4 Battery Implementation

We finally address the issue of how ants appear at each inverter's source, and disappear at each inverter's sink. Figure 7 illustrates the implementation of an ant-based battery and its parallels to an electronic battery.

The battery implementation relies on supplementing the circuit level with two additional levels, one above and one below the circuit level. The three levels communicate with each other through funnels of sufficiently narrow width so that only a single ant may pass through any given funnel at once. Ants falling into a funnel end up at the level just below the one at which they started. Initially, only the source level contains ants. Ants fall through the top funnels into the circuit level, at the source location of every inverter. Ants then move around in the circuit level until they reach an

3 It is known that every circuit is planar, meaning that for any given circuit it is always possible to construct a new circuit that computes the same function as the original one, and does not contain any wire-crosses. However, this new circuit is more complex and less natural than the original circuit, so that the use of wire-crosses is justified for simplicity reasons.

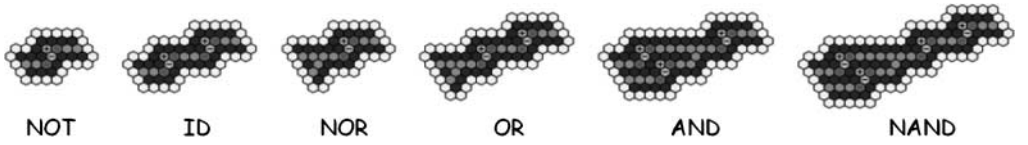

Figure 5. List of the basic logic gates.

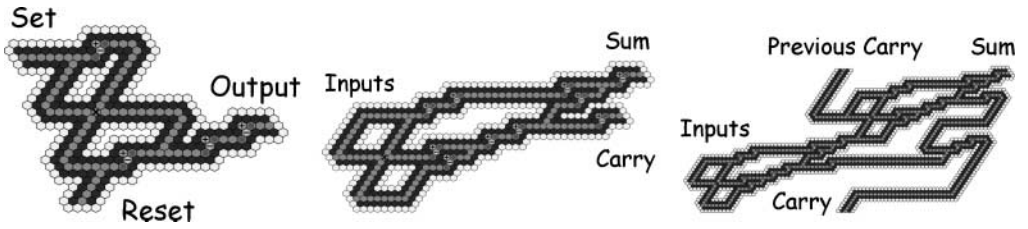

Figure 6. A one-bit memory, a half-adder, and part of a multi-bit adder.

inverter's sink location, at which point they fall through the bottom funnels into the sink level, where they stay from that point onward. The battery empties when the source level no longer contains enough ants to support a steady flow of ants into the circuit. By manually moving the ants from the sink level to the source level one effectively recharges the battery.

5 Discussion

In this section we elaborate on the intriguing parallels between ant-based and electronic circuits, and discuss some choices we have made in the design of the various ant-based primitive components.

5.1 Correspondence to Electronic Circuits

As has already been made clear throughout this work, the proposed model for ants and pheromones shares a lot of notions with electronic circuits. The correspondence is summarized in Table 2.

In this work we identify individual ants with individual electrons. A moving ant is therefore taken to correspond to a moving electron, and the movement of ants through paths can then be seen to correspond to the current flowing through a wire. In electronic circuits, current flows as a result of a difference in the potential between two points, with an electron probabilistically "choosing" to move toward areas with higher potential difference. Similarly, ants probabilistically choose to move toward higher pheromone concentrations. The similarities with electronic circuits extend beyond the simple correspondence of these notions. Without going into much detail, we outline these further similarities below.

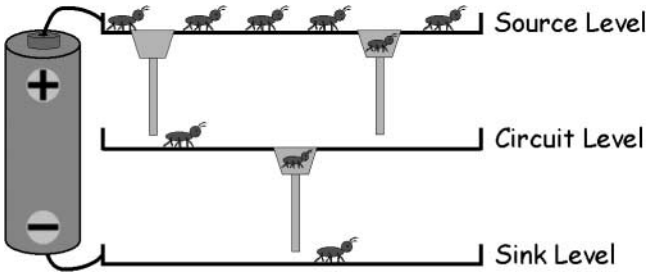

Figure 7. An electronic battery and an ant-based battery.

Table 2. Correspondence between ant-based circuits and electronic circuits.

Ant-based circuits	Electronic circuits
Ants	Electrons
Paths	Wires
Ant movement	Current flow
Sources, sinks	Battery poles
Pheromones	Potential
Gadgets	Logic gates

Ants in our model are assumed to move at a constant speed and without any obstacles, which leads us to equate ant flow with electrical current rather than voltage. This makes certain parallels with electronic circuits more involved than what one would hope for. For example, the problem of defining an inverter's noise regions is also present in electronic circuits, where it is usually stated with respect to voltage rather than current. Because of this difference, increasing the fan-out of a gate (i.e., the number of gates that it can drive) is much easier in electronic gates than in our case: Driving two gates does not require twice as much voltage in electronic gates, but it does require twice as much ant flow in ant-based gates.

Pushing the current-versus-voltage discussion a bit further, whereas electronic circuits are usually assumed to be connected to constant voltage sources, the ant-based circuits behave as if connected to constant current sources. The effect of splitting wires in the latter case is that current splits equally between the two wires, much like the ant flows, while in the former case one does not get such an effect.

For the interested reader, we note here that ant-based circuits also obey Kirchhoff's current law.[4] Combined, our assumption that ants move one step per time unit, and the fact that our circuits do not have dead ends, ensure that ants do not pile up at any location in the circuit, or, as Kirchhoff's current law states, that at each junction point the incoming flow equals the outgoing flow.

5.2 Alternative and Extended Designs

In designing the components of an ant-based computer, we have made several choices. We review some of these choices next and discuss possible alternatives and extensions of the proposed original design.

As was noted earlier, we assume that ants will not move backward within the circuits. Such a situation arises when ants enter a wire-merge from one of its inputs, and, instead of exiting from the designated output, they exit the wire-merge from its other input. This behavior effectively produces a phenomenon that corresponds to a short circuit. In electronic circuits this phenomenon is prevented by adding `ID` gates, which act as buffers, to the two incoming wires prior to their merging. Such an approach would also work in ant-based circuits, guaranteeing that ants moving backward would meet an inverter's sink and leave the system. Although this already happens in most cases using our current implementation, it is not always the case, as in circuits where the inputs are connected to an `OR` or a `NOR` gate. Hence, our design would benefit from enhancing all wire-merges with `ID` gates on their inputs, in order to avoid possible short circuits.

Wire-merges are the sources of even more problems, even when the problem above has been addressed. When two ant flows meet at a wire-merge, the merged flow is the sum of the two input flows. In particular, when the two input flows correspond to a logic `0`, we expect the merged flow to also correspond to a logic `0`, despite the doubling of the flow. As described in Sections 3.2 and 3.3,

4 There is also a Kirchhoff's voltage law, which does not, however, directly apply in the ant-based circuit setting.

the inverter can be designed and implemented in a way that accommodates such a demand. However, a solution to this problem seems to conflict with a solution to the output gain problem. Our experience with simulated circuits indicates that the model's parameters need to be carefully chosen in order to address both problems simultaneously.

We propose here an alternative implementation of the wire-merge that solves the problem of merging flows, hence leaving the inverter only with the output gain problem, which by itself is much easier to solve. In our alternative implementation, every wire-merge is followed by a wire-split, with one output flow directed to a sink, and the other to the intended destination of the merged flow. This effectively produces an `AVG` gate, where the output flow is the average of the two input flows. When both input flows correspond to a logic `0`, so does the output, without increasing the flow. In fact, this approach has the added benefit that when one takes the average of two input flows corresponding to a logic `1`, the output is still a flow in the interval [0, 1]; this is not necessarily the case in our original design. Lastly, inputs composed of a logic `0` and an amplified logic `1`, which can be readily obtained by using the `ID` gates proposed earlier, give an output of a logic `1`, as required. On the negative side, the use of an `AVG` gate introduces a second, in addition to the inverter, "computational" component into the design, while in our original design everything besides the inverter is essentially a wire.

Finally, recall that in the current design we rely on ant flows for signaling a logic `0` or a logic `1`. Alternatively, one can use pheromone concentrations, by observing that the difference between the pheromone concentrations at an inverter's point *Pump-B* and point *Sink* is a sufficiently large (positive) number when the inverter's output is a logic `1`, and a sufficiently small (negative) number when the inverter's output is a logic `0`. Hence, by terminating all circuits with an `ID` gate, and measuring the pheromone concentrations at the two aforementioned locations in the last inverter, one can obtain a reading of the circuit's output as well.

6 Conclusions and Future Work

We have shown that Markovian processes, such as those followed by ants, can collectively produce a coherent global behavior. We have illustrated this by proposing a simple and intuitive model of the behavior of ants and the physical properties of pheromones, and showing that the model is sufficiently rich to support universal computation. We have argued and shown through computer simulation that the proposed model possesses two properties that individually are presumably easy to obtain, but seem to be hard to accommodate simultaneously: the model is biologically and physically plausible, and at the same time it is expressive enough to make the construction of arbitrary logic circuits possible.

For future work one can pursue both further theoretical analysis and further experimentation. An obvious candidate for theoretical analysis is to solve the set of equations that describe the change in the state of an inverter and prove what we have empirically observed, namely, that these equations have a steady state for any given input to the inverter. In addition, one might try to compute the convergence time until a steady state is reached, which would correspond to the response time of the inverter. This can also be empirically measured and cross-checked with the theoretical bounds. Finally, one can derive bounds on the probability that the inverter's output is correctly computed for each input. Since each ant decides on its actions independently, there is a nonzero probability that a large number of ants will not behave as expected, and that the inverter will produce a bogus output. However, exactly because of the independence across ants, the probability of this event happening is extremely small. Although our existing empirical results already support this claim, it would be useful to also have an analogous theoretical result.

On the experimental side, one could explore other sets of parameters that make the inverter work, guided perhaps by theoretical results on what the relation between these parameters should be. Preliminary experimental results in this direction indicate that the proposed model is robust to parameter changes, as long as the parameters are changed in meaningful ways. So, for instance, when increasing the dissipation rate, one should presumably also increase the amount of pheromone entering the system. A second experimental direction would be to simulate an entire circuit, perhaps

a ring oscillator, a multi-bit adder, or a memory unit, and show that the circuit works as expected. A simulation that displays the circuit and the ants moving would provide an extremely interesting, and entertaining, visualization of the circuit computation. It is our belief that such a simulation might even prove useful as a pedagogical tool for introducing the concept of logic circuits and for illustrating that computation is a notion independent of the medium on which it is carried out.

Finally, from a biological point of view we would be greatly intrigued by the prospect that the behavior of any particular species of ants is sufficiently close to our proposed model so that a real ant-based computer could be implemented. Such a study lies, however, outside the scope of this work and is left to entomologists.

Acknowledgments

The author would like to thank his father, George Michael, for many hours of useful discussions on various aspects of this work, and especially for his help in clarifying the finer points in the parallels between ant-based circuits and electronic circuits. Useful feedback was also received from the anonymous reviewers.

References

1. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T. F., Nagpal, R., Rauch, E., Sussman, G. J., & Weiss, R. (2000). Amorphous computing. *Communications of the ACM*, *43*(5), 74–82.

2. Bonabeau, E., Theraulaz, G., & Deneubourg, J.-L. (1998). Fixed response thresholds and the regulation of division of labour in insect societies. *Bulletin of Mathematical Biology*, *60*(4), 753–807.

3. Deneubourg, J.-L., Aron, S., Goss, S., & Pasteels, J. M. (1990). The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, *3*(2), 159–168.

4. Deneubourg, J.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chrétien, L. (1991). The dynamics of collective sorting: Robot-like ants and ant-like robots. In J.-A. Meyer & S. W. Wilson (Eds.), *Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats* (pp. 356–363). Cambridge, MA: MIT Press.

5. Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge, MA: MIT Press.

6. Knight, T. F., & Sussman, G. J. (1998). Cellular gate technology. In C. Calude, J. Casti, & M. J. Dinneen (Eds.), *Proceedings of the First International Conference on Unconventional Models of Computation* (pp. 257–272). Berlin: Springer-Verlag.

7. Lumer, E., & Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. In D. Cliff, P. Husbands, J.-A. Meyer, & S. W. Wilson (Eds.), *Proceedings of the Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats* (pp. 501–508). Cambridge, MA: MIT Press.

8. Vestad, T., Marr, D. W. M., & Munakata, T. (2004). Flow resistance for microfluidic logic operations. *Applied Physics Letters*, *84*(25), 5074–5075.