THE ACHILLES PARADOX

URI ABRAHAM
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
BEN-GURION UNIVERSITY, BE'ER SHEVA, ISRAEL

1. Introduction

Those of you that were in our 1993 CS&P meeting in Nieborów (Poland) and heard my talk[1] can guess that I intend to speak about models for concurrency, and not to discuss a problem in philosophy. I should admit outright that since my reconstruction of the Achilles involves the notion of definability in the theory of models it cannot be a faithful rendering of Zeno's argument. However so little and indirect is our information about Zeno that the experts do not seem to be in agreement, and this may leave some place for a naive admirer.

I intend to formalize the paradox with system executions. These are mathematical structures (as defined by Tarski) that satisfy a certain finiteness property. (Lamport introduced system executions in [7] which has considerably influenced my work; his system executions, however, differ from those defined here in several aspects.) In my experience system executions are very useful to analyze concurrent systems, and I wish to exemplify this with the Achilles paradox. First I explain what system executions are, then I show how they are used to define concurrency, and finally I use them to analyze the paradox. Before let me discuss some of the "philosophical" assumptions that underly my choice of definitions.

Usually events have positive (i.e., > 0) duration. Yet even in everyday experience we often associate a single number to record the time of an event, rather than a pair of numbers to indicate an interval. For example, we say "Achilles finished the race at $14 : 34 : 21$" to record his arrival time. We may strive for higher precision, and we believe that it is possible to obtain better accuracy because we think that there is a single number that represents this event. To exclude durationless events from our theory, i.e. to accept only events with non-zero duration, seems too severe a limitation. Henceforth we do accept durationless events, and call them "instantaneous events". The time of an instantaneous event e is denoted $time(e)$.

$\prec$ denotes the precedence relation on the events. (Intuitively, event a precedes event b if a ends before a begins.) It is not a linear ordering since two events may overlap, but the restriction of $\prec$ to the instantaneous events *is* linear and

$$e_1 \prec e_2 \text{ iff } time(e_1) < time(e_2).$$

Here $<$ is the familiar ordering on the real numbers.

A usual assumption in the realm of computer science is that the set of events that precede any given event is finite. That is, for any event e the set

$$\{x \mid x \text{ is an event and } x \prec e\}$$

Date: June 11, 2001.
[1]Bakery Algorithms.

is finite. We accept this finiteness demand even for instantaneous events.

For a computer scientist, used to identify events with executions of instructions from some program, this finiteness condition is very natural and helpful since it allows proofs by induction on the size of the set of predecessors of an event.

Only terminating events are considered in this paper, that is events with a finite duration. Lamport [7] introduced and used a stronger finiteness condition: not only that the set of predecessors of e is finite, but so is also the set of those events that are $\prec$-incomparable with e. That is, for every (terminating) event e there are only finitely many exceptions to those that satisfy the formula $e \prec x$.

It follows immediately from our discussion that the set of instants

$$\{\mathit{time}(e) \mid e \text{ is an instantaneous event}\}$$

is at most of order-type ω (the order type of the natural numbers). So surely there are real numbers that are not of the form $\mathit{time}(e)$. These numbers are mathematical objects, not corresponding to any "real" event.

It makes sense however to assume that every *definable* instant is of the form $\mathit{time}(e)$ for some instantaneous event e. That is, definable instants have a higher existential status under this assumption. Accepting this view puts a severe limitation on the language in which these instants are defined, for if the language is too rich then there would be definable instants with no corresponding instantaneous events. For example, if division by 2 is allowed, then the so called "dichotomy" argument shows that there must be definable instants with no corresponding events. Indeed, assuming the existence of two instantaneous events with times $t_0 < t_1$, define first $(t_0 + t_1)/2$ and continue to define the mid points until you reach an instant with no corresponding event (for there are only finitely many events preceding the event of t_1). It can be argued however that "division by 2" is not a natural operation in this context, since the fact that both of t_0 and t_1 are remarkable moments does not say anything about their mid point. The Achilles paradox can be presented as an argument that, even in a very simple language with no arithmetical operations, concurrency brings about a definable eventless instant.

We are used today to represent both time and space with real numbers, and by this we implicitly declare quite a strong statement about the nature of time and space, namely their isomorphism. Since the argument of Zeno in the Achilles does not assume this time–space correspondence, we will not impose it and use two sorts R_χ and R_τ to represent time and location (chronos and topos).

2. System executions and concurrency

A system execution is a structure for some signature in which the finiteness condition holds. We will explain this and give as example a system execution that describes a single running of Achilles. Similar system executions can be used to describe runnings (if I may use this term) of the tortoise (alone, without Achilles at her side), and then the combination of an Achilles system execution and a tortoise system execution is a concurrent run by both Achilles and the tortoise. It is on this structure that Zeno's paradox will be conducted. (Of course, a real race with all the excitement around is never a mere concurrent combination of separate runs of the participants, but this is a different issue.)

To define a (first-order) language we need to know what are the special symbols used in the language (its *signature*). As we are taught in elementary logic courses, a *signature* is a list of function symbols, predicate symbols and symbols called constants, where to each function and predicate symbol a positive natural number is assigned (its arity). A *structure* for a signature L is an arbitrary universe and interpretation for the symbols of L, respecting their arities. For example, a function symbol of arity two is interpreted by a two-place function. This should be clarified by the example given below.

We use here multi-sorted rather than single-universe structures. The universe of a multi-sorted structure consists of pairwise disjoint *sorts*, corresponding intuitively to the different types of objects we deal with. Accordingly, the first-order language contains distinct sets of variables, one for each sort. This enables shorter expressions. Also, a multi-sorted signature gives more information: Not only the sorts are enumerated separately, but with each function symbol two sorts are specified, its domain and range sort. Similarly each predicate symbol has its domain sort, and each constant is from a specific sort.

The following defines a signature, called L_A (A is for Achilles), which we use to describe Achilles runs.

1. Three sorts: E (for events), R_χ (for instants), and R_τ (for locations). (At this stage, E, R_χ and R_τ are just symbols.)
2. A binary relation symbol $\prec$. Its domain is specified to be E. $\prec$ is called the "precedence" relation on the events. (When we say that the domain of $\prec$ is E, we mean that the semantics of the formula $x \prec y$ imply that both of x and y are in E.)
3. Two binary relation symbols $<_\chi$, and $<_\tau$. Their domains are R_χ and R_τ respectively. However, for simplicity of expression, we will write $<$ instead of $<_\chi$ or $<_\tau$ when no confusion is possible. [The intention is to use $s < t$ when instant s is earlier than t, and to use $u < v$ for locations u and v when u is nearer than v to the starting point of the race.]
4. A unary predicate symbol I, defined over sort E. (Our intention is to use $I(x)$ for "x is an instantaneous event".)
5. A unary function symbol *time*. Its domain is sort E and its range is sort R_χ. (The value *time*(e) is called "the instant of e". Of course, it is only meaningful if e is instantaneous, and yet *time* is required to be defined on all events, even if not instantaneous. This problem comes from the requirement that functions are total on their sort, and often a special "undefined" value is used to enable total functions to mimic partial ones.)
6. A unary function symbol d_A has domain R_χ and range R_τ. It is called "the location function" and our intention is that $d_A(t)$ gives the location of Achilles at instant t.
7. Two constants t_0^A and t_1^A are specified to be in R_χ. (t_0^A will be used to denote the time Achilles starts his course, and t_1^A is the time he arrives to its end.)

We say that $\mathcal{M}$ is an interpretation (or structure) for L_A iff $\mathcal{M}$ is a function defined on the symbols in L_A such that:

1. $\mathcal{M}(E)$, $\mathcal{M}(R_\chi)$, and $\mathcal{M}(R_\tau)$ are non empty pairwise disjoint sets. (Members of $\mathcal{M}(E)$ are called events of $\mathcal{M}$, members of $\mathcal{M}(R_\chi)$ are called instants, and members of $\mathcal{M}(R_\tau)$ are called locations.)
2. $\mathcal{M}(\prec)$ is a binary relation on $\mathcal{M}(E)$.
3. $\mathcal{M}(<_\chi)$ is a binary relation on $\mathcal{M}(R_\chi)$, and $\mathcal{M}(<_\tau)$ is a binary relation on $\mathcal{M}(R_\tau)$.
4. $\mathcal{M}(I)$ is a unary relation on $\mathcal{M}(E)$.

5. $\mathcal{M}(time) : \mathcal{M}(E) \longrightarrow \mathcal{M}(R_\chi)$. That is, $\mathcal{M}(time)$ is a function assigning instants to events.
6. $\mathcal{M}(d_A) : \mathcal{M}(R_\chi) \longrightarrow \mathcal{M}(R_\tau)$.
7. $\mathcal{M}(t_0^A), \mathcal{M}(t_1^A) \in \mathcal{M}(R_\chi)$.

In lectures and informal writing it is customary to omit $\mathcal{M}$ and to write, for example, E instead of $\mathcal{M}(E)$ unless confusion occurs.

Since we are interested in structures that represent possible runs of Achilles, certain interpretations may be discarded at once. For example, those in which $<$ is not a linear ordering on R_χ. The collection of all "acceptable" structures for L_A will be denoted S_A, but how are we going to define it? There will be two tests an L_A-structure must pass in order to be included in S_A: a first-order and a second-order test. The first-order test is a list of axioms Th_A that the structure must satisfy, and the second-order test is the finiteness condition defined below. Any structure that passes these tests is called "a system execution for Th_A", and S_A is the class of all system executions for Th_A; it is called the *system* for Th_A.

Definition 2.1 Th_A IS THE FOLLOWING FINITE SET OF FIRST-ORDER SENTENCES IN THE LANGUAGE OF L_A (FOR SIMPLICITY SOME OF THESE SENTENCES ARE GIVEN IN ENGLISH).

1. $\prec$ IS A PARTIAL ORDERING ON E SUCH THAT
$$\forall a, b, c, d \in E \ (a \prec b \wedge \neg c \prec b \ \wedge c \prec d \longrightarrow a \prec d).$$
[AN EXPLANATION OF THIS "INTERVAL ORDERING" FORMULA CAN BE FOUND IN [5], [6], AND [7] .]
2. $<_\chi$ AND $<_\tau$ ARE IRREFLEXIVE LINEAR ORDERINGS ON R_χ AND R_τ RESPECTIVELY. [WE WILL USE THE PLAIN SYMBOL $<$ FOR BOTH $<_\chi$ AND $<_\tau$, AND, AS USUAL, $x \leq y$ IS A SHORTHAND FOR $x < y \vee x = y$.]
3. $t_0^A < t_1^A$, AND THE FUNCTION d_A IS MONOTONICALLY INCREASING ON THE INTERVAL $[t_0^A, t_1^A]$. THAT IS
$$\forall r, s \in R_\chi \ (t_0^A \leq r < s \leq t_1^A \longrightarrow d_A(r) < d_A(s)).$$
4. $\forall e_1, e_2 \in E \ (I(e_1) \wedge I(e_2) \longrightarrow ((e_1 \prec e_2) \text{ IFF } time(e_1) < time(e_2)).$

[THAT IS, FOR ANY TWO INSTANTANEOUS EVENTS THE PRECEDENCE RELATION $\prec$ AND THE $<$-ORDERING OF THEIR TIMINGS AGREE.]

Now we state the additional second-order requirement. If $\mathcal{M}$ is any model of Th_A (a structure that satisfy all the axioms in Th_A), then we say that $\mathcal{M}$ is a "system execution for Th_A" iff the relation $\prec_{\mathcal{M}}$ satisfies the following finiteness demands. For every event e in $\mathcal{M}$:

1. Only finitely many members of E are $\prec$ incomparable with e in $\mathcal{M}$. That is, except for a finite number of events in $\mathcal{M}$, all others satisfy the formula
$$x \prec e \vee e \prec x$$
2. The set of events x in $\mathcal{M}$ that satisfy $x \prec e$ is finite.

Combining these two demands we get: Except for finitely many events $e \prec x$ holds for all other events. As mentioned above, this finiteness condition was introduced and used by Lamport.

2.1. **Concurrency.** The collection of all possible runs of the tortoise can be similarly defined. First we set the language by defining a signature L_T obtained by exchanging "A" with "T" in the symbols of L_A. That is, L_T contains

1. Three sorts: E, R_χ, and R_τ.
2. A binary relation symbol $\prec$. Its domain is specified to be E.
3. Two binary relation symbols $<_\tau$ and $<_\chi$ with domains R_τ and R_χ.
4. A unary predicate symbol I, defined over sort E.
5. A unary function symbol *time*. Its domain is sort E and its range is sort R_χ.
6. A unary function symbol d_T has domain R_χ and range R_τ. [It is called "the location function for the tortoise" and our intention is that $d_T(t)$ gives the location of the tortoise at time t.]
7. Two constants t_0^T and t_1^T are specified to be in R_χ. [t_0^T is used to denote the time the tortoise starts her course, and t_1^T is the time she arrives to the end.)

Now, Th_T is defined just as Th_A with the tortoise symbols instead of Achilles'. Then S_T is the collection of all system executions that satisfy the axioms of Th_T.

Intuitively then S_A and S_T represent the systems of all possible runs of Achilles and of the tortoise (separately).

Now we treat concurrency. What does it mean that both Achilles and the tortoise run concurrently? Disregarding possible psychological affections, we represent such a concurrent run as an independent combination of a structure from S_A with a structure from S_T. Let us be more precise and define first a signature $L = L_A \cup L_T$. Specifically, L contains each of the symbols that appear in both L_A and L_T (such as *time*) with their specified arity, and then L contains d_A as well as d_T, and the four constants t_0^A, t_0^A etc. Now for any structure $\mathcal{M}$ for L two structures (reducts) will be defined below: $\mathcal{M}\restriction L_A$ an interpretation of L_A, and $\mathcal{M}\restriction L_T$ an interpretation of L_T. Then the concurrent system S is defined as the class of those structures $\mathcal{M}$ for which $\mathcal{M}\restriction L_A \in S_A$ and $\mathcal{M}\restriction L_T \in S_T$.

For example to define restriction to L_A let $\mathcal{M}$ be any structure for L, then define $\mathcal{N} = \mathcal{M}\restriction L_A$ by taking the universe of $\mathcal{M}$ with its three sorts: $\mathcal{N}(E) = \mathcal{M}(E)$, $\mathcal{N}(R_\chi) = \mathcal{M}(R_\chi)$, and $\mathcal{N}(R_\tau) = \mathcal{M}(R_\tau)$. Then $\mathcal{N}$ interprets all the symbols in L_A exactly as they are interpreted by $\mathcal{M}$. To say it in other words, $\mathcal{N}$ is obtained by throwing away from $\mathcal{M}$ all interpretations of symbols not in L_A.

Similarly, Π_T is defined to restrict interpretations of L to interpretations of L_T. Note that the events in $\mathcal{M}$ pertaining to Achilles remain in $\mathcal{M}\restriction L_T$ (since both structures have the same universe), but now they have "lost their meaning". (Restrictions can be defined more generally whenever L_1 is a subsignature of L_2.)

Let us define now

$$S = S_A \mid\mid S_T = \{\mathcal{M} \mid \mathcal{M} \text{ is a structure for } L \text{ such that } (\mathcal{M}\restriction L_A \in S_A \text{ and } \mathcal{M}\restriction L_T \in S_T\}.$$

S is the system of all "concurrent executions" of Achilles and the tortoise. However, if $\mathcal{M} \in S$, then it is not necessarily a race: the point is that in a race two additional requirements must be satisfied.

Definition 2.2 We say that $\mathcal{M} \in S$ is a *race* iff

1. $t_0^A = t_0^T$ holds in $\mathcal{M}$. That is, Achilles and the tortoise start the race at the same time. We shall denote with t_0 this common time.
   $d_A(t_1^A) = d_T(t_1^T)$ holds in $\mathcal{M}$. That is, both the tortoise and Achilles arrive to the same final location. (This is a natural demand, but it will never be used.)

2.
$$(t_1^T < t_1^A) \vee \exists t\,[(t_0 < t \leq t_1^A) \wedge d_A(t) = d_T(t) \wedge \forall s\,(t_0 \leq s < t \longrightarrow d_A(s) < d_T(s)].$$

That is, either the tortoise wins because she reaches the end of the course before Achilles does, or else there is an instant t, sometimes after the beginning but before (or at) the end of the race, in which Achilles "touches" the tortoise for the first time: At each earlier instant the tortoise is ahead of Achilles (and in particular, then, at t_0, the tortoise is given a head start over Achilles).

Define R as the set of all "races", i.e., the set of all system executions in S that satisfy the two statements above.

We have defined the class R indirectly: We first defined systems S_A and S_T, and then their concurrent composition S, and finally only those system executions in S that are races were taken in R. We did so to illustrate the concept of concurrent composition. However it may be easier to define R directly by first defining the joint signature L and then stating axioms $Th_A \cup Th_T$ together with the two additional "race" conditions, and taking all those system executions that satisfy these axioms. Anyhow, we say that the tortoise wins in $\mathcal{M}$ if $t_1^T < t_1^A$ holds, but otherwise Achilles is declared the winner.

2.2. **Definability.** The notion of definability has a central place in our interpretation of the Achilles paradox, and so we recall what it means for an individual (a member of the universe of a structure) to be definable. Let L be some signature and $\mathcal{M}$ a structure for L.

Definition 2.3 We say that a formula $\phi(x)$ (in the language of L and with a single free variable x) is a *definition* in $\mathcal{M}$ iff the following sentence holds:
$$\exists! x \phi(x).$$
The symbol $\exists! x$ means as usual "there is a single x such that..." So, if $\phi(x)$ is a definition in $\mathcal{M}$ then there is a single individual a such that $\mathcal{M} \models \phi[a]$. This unique a in $\mathcal{M}$ is said to be "definable" by $\phi(x)$, and we say that a is $\mathcal{M}$-definable.

Let us return now to the signature $L = L_A \cup L_T$ defined previously and to the system S of all concurrent executions as defined above. We say that $\mathcal{M} \in S$ satisfies the *temporal existence property* iff for every definition $\phi(x)$ in $\mathcal{M}$, and individual a in $\mathcal{M}$ that is definable by ϕ, if $R_\chi(a)$, then there is an event e in $\mathcal{M}$ such that $I(e) \wedge a = time(e)$.

Intuitively then, the temporal existence property says that any instant that can be defined "really exists": a definable instant is necessarily experienced as the exact time of an event. Our point of view in analyzing Achilles is that this is a reasonable property because it allows to distinguish between those members of R_χ that are there just for theoretical completeness and those that have a "good reason" to exist. Anyhow, the temporal existence property is just a property of system executions, and our effort to attach a meaning to it has no effect on the argument and its mathematical content.

Another definition is needed before we tackle the paradox itself. The image of a continuous function on a closed real interval is, as we all know, a closed interval contained between the maximum and minimum of the function. In particular, the intermediate value property holds for any continuous function: If $f(x_1) < y < f(x_2)$ then for some x between x_1 and x_2, $f(x) = y$. Functions in physics are usually assumed to be continuous, and hence they satisfy the intermediate value property. I don't know if the Greeks would accept continuity without hesitation, and if time is assumed to be discrete, then continuity is meaningless. Yet the intermediate value property of a physical function seems to be "evident". For example, if Achilles is at location d_1 at some moment and then at some farther location d_2 at a later moment, then surely he passes through every intermediate location. However we present Zeno's paradox as a refutation of this property, and of even a weaker one.

Definition 2.4 Let $\mathcal{M}$ be a system execution in S. We say that d_A satisfies the *weak* intermediate value property in $\mathcal{M}$ (in the temporal interval $[t_0^A, t_1^A]$) iff whenever a, in $\mathcal{M}(R_\tau)$, is $\mathcal{M}$-definable and satisfies $d_A(t_0^A) \leq a \leq d_A(t_1^A)$ in $\mathcal{M}$, then for some instant t in the interval $[t_0^A, t_1^A]$ in $\mathcal{M}$, $d_A(t) = a$ holds. In other words, the weak intermediate value property says that even if the range of d_A omits some values, it obtains all those that are definable in $\mathcal{M}$ in the interval between $d_A(t_0^A)$ and $d_A(t_1^A)$ (the two extremes of this monotonic function in $\mathcal{M}$).

3. Zeno's second paradox

The usual description of the Achilles is to assume constant speeds for Achilles and for the tortoise, to fix a head start that puts the tortoise in front of Achilles, and then to define an infinite, increasing sequence of instants $\langle t_i \mid i = 0, 1, \dots \rangle$ such that for each i the tortoise at t_i is strictly between Achilles and the finish post. To begin with, t_0 is the starting time of the race. If t_i is defined and $l_i < l'_i$ are the locations of Achilles and the tortoise respectively, at t_i, then t_{i+1} is computed as the time by which Achilles reaches l'_i. The final argument derives a contradiction from the contrast between this non terminating description and the termination of the race.

I wish to present the argument in the simplest possible first-order language without using induction, and I choose a somewhat different path of reasoning, which allows the whole argument to be conducted as a first-order proof.

Theorem 3.1 (Zeno)**.** *Suppose that $\mathcal{M}$ is a race that satisfies the temporal existence property. If Achilles wins, then the location function d_A in $\mathcal{M}$ does not satisfy the weak intermediate value property.*

Proof. Suppose that Achilles wins the race $\mathcal{M}$. This means by definition that $t_1^A \leq t_1^T$ holds. Thus by (2) in Definition 2.1 there exists an instant t' such that the following holds in $\mathcal{M}$

$$t_0 < t' \leq t_1^A \wedge d_A(t') = d_T(t') \wedge \forall s(t_0 \leq s < t' \longrightarrow d_A(s) < d_T(s)).$$

We shall denote by f the location $d_A(t') = d_T(t')$ in $\mathcal{M}$, where Achilles "touches" the tortoise for the first time. Observe that t' is definable because there is at most a single individual t' in $\mathcal{M}$ as above. Hence by the assumed temporal existence property for some instantaneous event called "touch_event",

$$t' = time(touch\_event).$$

Let i be a last instantaneous event in $\mathcal{M}$ such that $i \prec touch\_event$; i.e. such that there is no instantaneous event e with $i \prec e \prec touch\_event$. Why is there such an event i? Observe first that $\mathcal{M}(t_0)$ (the interpretation of t_0) is definable in $\mathcal{M}$ by the formula $x = t_0$. Hence there is an instantaneous event i_0 with $t_0 = \mathit{time}(i_0)$. Hence the set

$$D = \{e \mid I(e) \ \wedge \ i_0 \preceq e \prec \mathit{touch\_event} \text{ in } \mathcal{M}\}$$

is nonempty. By the finiteness property D is finite, and hence possesses an $\prec$-maximal element i. (Such i is not necessarily unique, because we did not exclude the possibility of two instantaneous events with the same time.)

Now define $t = \mathit{time}(i)$ in $\mathcal{M}$. Then $t_0 \leq t < t'$ and t is definable as the $<$-maximal instant t in $\mathcal{M}$ such that

1. $t < t'$,
2. there exists an instantaneous event i such that $t = \mathit{time}(i)$.

Since t is $\mathcal{M}$-definable, $m = d_T(t)$ is definable as well. Now $m < d_T(t') = f$ follows from the monotonicity of d_T and the relation $t < t'$.

Claim 3.2. *Let $k = d_A(t)$. Then $k < m$ in $\mathcal{M}$. (That is, at time t, Achilles is still behind the tortoise).*

Proof. This follows from the assumption that

$$\forall s(t_0 \leq s < t' \longrightarrow d_A(s) < d_T(s)).$$

To sum up, we have seen that $d_A(t) = k < m < f = d_A(t')$ holds in $\mathcal{M}$, where m is a definable location. We claim that m proves that d_A does not have the weak intermediate value property: There is no instant s such that $t < s < t'$ and $d_A(s) = m$. Why? For if there were such an instant s in $\mathcal{M}$, then it would be unique by monotonicity of d_A, and hence $\mathcal{M}$-definable (see below). But then, by the temporal existence property, there would be an instantaneous event e with $s = \mathit{time}(e)$. But this would contradict the maximality of t!

The formula that would define s is the following formula with a free variable x, in which all parameters are definable:

$$t < x < t' \wedge m = d_A(x).$$

Though system executions were defined with the second-order finiteness property, the reader may have remarked that we only used a first-order property in the proof: That for every event, if it is preceded by an instantaneous event, then it is preceded by a last (i.e. $\prec$-maximal) instantaneous event. In general, the finiteness property can be replaced by a scheme of first-order axioms, and these were sufficient in all the applications that I've seen.

4. Conclusion

You may wonder what all of this has to do with computer science and whether the engineer and theoretician may gain by considering system executions. Most of the approaches to concurrency in vogue today, so it seems to me, give to the notion of a state a central place. The alternative, to base the theory of concurrency on events and the precedence relation, was considered and dismissed, too early in my opinion, as being inadequate for a formal theory. I believe that more experience is needed in order to find the place of model theoretic considerations such as those presented here.

For example, suppose you have to design the strategy of a missile pursuing in space a slower craft in order to meet it. There are different possible strategies, and the following one resembles Achilles'. The pursuer assesses very quickly the position of the spacecraft, and then gets to this position at full speed. The spacecraft, however, has meanwhile moved away, and so this recursive procedure is repeated until the two engines meet. Probably you have better strategies in mind and perhaps you want to compare them with this most simple one. Anyhow, I believe that in describing such strategies and in arguing about them, system executions may be valuable and natural tools.

The Achilles paradox is given as an example. You may disagree with my interpretation or try to construe the paradox differently; my point is that model theory is relevant for this type of discussion. If you agree to the soundness of my choice of tools to analyze Zeno, then you may be willing perhaps to consider the relevance of these tools for computer science problems as well, and if so I would like to suggest for your consideration the work in the References and some still unpublished lecture notes on concurrency.

Acknowledgements. To James Cummings Mati Rubin and John Tromp for valuable comments.

References

1. Abraham, U. 1995. On interprocess communication and the implementation of multi-writer atomic registers. *Theoretical Computer Science* 149, pp. 257-298.
2. Abraham, U. 1993. What is a State of a System (an outline). In M. Droste and Y. Gurevich eds, *Semantics of Programming Languages and Model Theory*, Volume 5 of Algebra, Logic and Applications, pp. 213–243, Gordon and Breach.
3. Abraham, U. 1993. On system executions and states. *J. Applied Intelligence* 3, pp 311–323.
4. Abraham, U. 1994. Bakery Algorithms. In H-D. Burkhard, L. Czaja, and P. Starke, eds, *Proceedings of the Concurrency, Specification and Programming Workshop, Nieborów 1993*, pp. 7–40, Wydawnictwa Uniwersytetu Warszawskiego, Warszawa.
5. Abraham, U., Ben-David, S., and Magidor, M. 1990. On Global-Time and Inter-process Communication. In *Semantics for Concurrency, Proceedings of the International BCS-FACS Workshop, Leicester* (edited by Kwiatkowska et al, Springer-Verlag in collaboration with the British Computer Society), pp. 311 – 323.
6. Fishburn, P. C. 1986. *Interval orders and interval graphs.* Wiley (Wiley-Interscience series in discrete mathematics).
7. Lamport, L. 1986. On Interprocess Communication, Part I: Basic formalism, Part II: Algorithms. *Dist. Comput.*, *1*, 77 – 101.