Institute of Computer Science
Academy of Sciences of the Czech Republic

Relativistic Computers and Non-Uniform Complexity Theory

Jiří Wiedermann and Jan van Leeuwen

Technical report No. 866

April 2002

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: (+4202) 6605 3520, fax: (+4202) 865 85 789,
e-mail:jiri.wiedermann@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Relativistic Computers and Non-Uniform Complexity Theory[1]

Jiří Wiedermann[2] and Jan van Leeuwen[3]

Technical report No. 866

April 2002

Abstract:

Recent research in theoretical physics on 'Malament-Hogarth space-times' indicates that so-called relativistic computers can be conceived that can carry out certain classically undecidable queries in finite time. We observe that the relativistic Turing machines which model these computations recognize precisely the Δ_2-sets of the Arithmetical Hierarchy. In a complexity-theoretic analysis, we show that the (infinite) computations of $S(n)$-space bounded relativistic Turing machines are equivalent to (finite) computations of Turing machines that use a $S(n)$-bounded advice f, where f itself is computable by a $S(n)$-space bounded relativistic Turing machine. This bounds the power of polynomial-space bounded relativistic Turing machines by TM$/poly$. We also show that $S(n)$-space bounded relativistic Turing machines can be limited to one or two relativistic phases of computing.

What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics.
in: D. Deutsch, The Fabric of Reality, 1998.

Keywords:
relativistic computing, relativistic Turing machines, non- uniform complexity, computability

[1]This research was partially supported by GA ČR grant No. 201/02/1456 and by EC Contract IST-1999-14186 (Project ALCOM-FT).

[2]Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic

[3]Institute of Information and Computing Sciences, Utrecht University, Padualaan 14, 3584 CH Utrecht, the Netherlands, email: jan@cs.uu.nl

1 Introduction

A number of phenomena in science have lead to an increased interest in models of computation that are more powerful than classical Turing machines. The phenomena range from evolving systems in biology to non-classical time-models in (theoretical) physics. Whereas these contexts lead to many new questions, it is of interest to study the computational theory of these phenomena in relation to classical complexity theory and its ramifications. In this paper we explore the computational limits and features of the so-called 'relativistic computers' that were recently proposed by Etesi and Németi [7], a kind of computers based on considerations from general relativity theory.

Relativistic computers will be seen to belong to the category of machine models whose powers transcend those of classical Turing machines and thus go beyond the range of the Church-Turing thesis. Models of computation that do not obey the Church-Turing thesis have been investigated since Turing's times (cf. [14]). The existence of such models in the realm of mathematics is already sufficient reason for their investigation. For example, Turing machines with oracles [16] and Turing machines with a restricted type of oracles known as *advice* [12] are at the heart of relativised and non-uniform computational complexity theory, respectively (cf. [1, 9]).

In newer studies, there is an increasing number of models that are justified by some physical reality in which they exist or could exist. Among the recent examples are the *evolving interactive systems*, which model the adaptive and potentially infinite computing behavior of Internet-like networks and of other 'living systems', real or artificial [20]. The model has been shown to possess 'super-Turing' computational power (cf. [17], [18]), and illustrates that realistic instances of computational systems can be identified that have this property. Further examples and related studies appear in e.g. [3, 4, 19].

Independently of this, the quest for computing beyond Turing machine limits has gained interest in theoretical physics. Namely, it has been shown that certain *relativistic space-times theories* license the idea of observing the infinity of certain discrete processes in finite physical time [7]. It is speculated that a similar phenomenon could occur in the realm of quantum computing as well [5]. From the viewpoint of computability theory the resulting *relativistic computers* can be seen as Turing machines with a suitable oracle, but it is the fact that these computers are based on apparently realistic 'thought experiments' and also the nature of these experiments that gives them a special status. The consequences of this observation for the foundations of computing have begun to occupy philosophers and physicists ([7],[10]), and recently mathematicians and logicians have begun to investigate even more general models of infinite-time computations [8].

In this paper we aim to appraise the power of *infinite computations* as realized by *relativistic computing* from the viewpoint of relativised and non-uniform computing. In this sense this paper complements the study in [7] where only some consequences for computability theory are discussed. Here we extend the results of [7], after defining a suitable model of *relativistic Turing machines* in Section 2. We observe that relativistic computing has precisely the power of recognizing the Δ_2-sets of the Arithmetical Hierarchy [14]. If the underlying physical theory is accepted, this would lift the barrier of recursiveness 'in nature' to the Δ_2-level, i.e. without violating any feasible thought experiments in general relativity theory. In Section 3 we give a complexity-theoretic characterization of relativistic computing in terms of a far simpler model from computational complexity theory: Turing machines with advice. We prove that under mild assumptions, $S(n)$-space bounded relativistic Turing machines are equivalent to 'classical' Turing machines with $S(n)$-bounded advice, with the further proviso that the advice is relativistically $S(n)$-space bounded computable. Given the physical background, the required number of relativistic phases of computing is a crucial resource. We show that any language that is relativistically recognizable within $S(n)$-space, can be recognized by a $S(n)$-space bounded relativistic Turing machine that needs at most 2 relativistic phases.

We emphasize that relativistic computing models are only claimed by certain theories in contemporary theoretical physics. The validity of the model has not been experimentally verified.

2 Relativistic and Other Computational Models

Our main goal is to compare the *infinite computations* performed by relativistic computers [7] with those performed in finite time by Turing machines *with advice.* In this section we describe the formal models of computations involved and give some basic constructions, cross-pointing to known concepts in recursion theory.

2.1 Relativistic Turing machines

Recent research in theoretical physics suggests that in certain relativistic space-times, viz. in Malament-Hogarth space-times, it may be possible for a computer to receive the answer to a *yes or no question* from an *infinite*, i.e. arbitrarily long computation in finite proper time (cf. [6, 7, 10]). Our first goal is to cast relativistic computations in the precise framework of Turing machines with oracles [14].

Informally, a *relativistic Turing machine* (RTM) is a kind of multitape deterministic Turing machine with a separate read-only input and, in the case of transducers, a write-only output tape. That is, we will consider RTMs both as language recognizers and as transducers, computing partial functions that map finite strings of input symbols to similar strings of output symbols. A RTM has a number of special, distinguished states that allow the implementation of *relativistic phases* of computing. More formally, a RTM is a seven-tuple $R = (\Sigma, Q, \delta, q_0, q_F, q_R, S)$ where Σ is a finite alphabet, Q is a finite set of states, δ is a standard (deterministic) transition function, q_0, q_F, and $q_R \in Q$ are the distinguished *initial, final* and *relativistic states*, respectively, and $S \subset Q - \{q_0, q_F, q_R\}$ is the set of *signal states.* Set S consists of a distinguished, so-called *no-signal state* q_N, and one or more *yes-signal states* q_Y.

A *relativistic computation* by a RTM R proceeds as follows, modeling the features described in [7]. On a finite input w, R starts its computations as a standard Turing machine in state q_0. It reads symbols from the input tape, prints symbols to the output tape, rewrites the working tapes, enters new states from $Q - S - \{q_R\}$, etcetera, and continues in the 'classical' way of computing until it enters its relativistic state q_R. At this moment R starts a *relativistic phase* in its computation. R now proceeds like an oracle Turing machine that executes an oracle call, with noted differences in the way the 'relativistic call' is elaborated *and in the way information is extracted from a single call.* Informally, entering q_R is like spawning a copy of the computation and asking: *starting from the current instantaneous description and continuing computing in accordance with my transition function, will my copy ever generate a yes-answer?* If so, then a yes-state $\in S$ is entered *after finite time*, with the state corresponding to the specific yes-signal which was emitted. (In [7] the finite moment is referred to as a 'Malament-Hogarth event'.) Otherwise, i.e. if no yes-signal is reached by the time of the Malament-Hogarth event, the no-signal state q_N is entered.

In *normal relativistic mode* the spawning process just waits ('observes') until a signal state is entered. All data on the machine's tape remain as they were when entering q_R, and also the heads retain their previous positions. As soon as a signal-state is reached, necessarily after finite time, the present relativistic phase ends, even though the infinite computation that was spawned may still be continuing. In *extended mode* the spawning process can observe a finite number of yes-signals from the computation, counting signals in a buffer of finite size and going from yes-signal state to yes-signal state until either the buffer is full or it 'sees' by the time of the Malament-Hogarth event that no further yes-signals will come (in which case it will transfer to q_N). In either case the relativistic phase ends and R resumes its computation from whatever signal-state it reached. R cannot switch to relativistic phases during a relativistic phase.

The description is only complete if it is specified when and what yes-signal is to be generated during a relativistic phase, from a given finite set of possibilities. Following [7] we assume that a yes-signal is triggered whenever some observable property P holds for the (instantaneous description of the) computation. It means that R must be augmented with a mechanism for checking P in a relativistic phase, i.e. during the spawned computation as it goes to infinity. We allow the mechanism to output signals from a finite set of different yes-answers, corresponding to the different signal-states. In [7] the simple case with $P =$ '*halting*' is considered, which only needs an easy program. The resulting mechanism will be termed a *relativistic oracle* for P.

After a RTM has resumed its classical mode of computation, it can switch to a relativistic phase again (i.e. when q_R is entered again). Thus, a computation of a RTM alternates between classical and relativistic phases of computing. A classical phase begins in a state from $Q - \{q_R\}$ and ends in q_R or q_F. A relativistic phase begins in state q_R and ends in a signal state $\in S$ and, in extended mode, with a value in a buffer of finite size. A RTM can halt only in state q_F. If it works as a transducer, the result of the machine's computation is written on its output tape. We claim:

Postulate A *Relativistic computations are faithfully modeled by computations of ordinary Turing machines using relativistic oracles in normal or extended mode.*

In [7], [21] the underlying assumptions are explained, viz. the observability of spawned infinite processes in finite physical time. Formally, our analysis shows that relativistic oracles are just oracles for 'eventually P' *except* when extended mode is used. The latter mode was not explicitly distinguished in [7] but will be important.

It is not entirely clear whether repeated relativistic phases of computation during the same computation of a RTM are physically feasible. The analysis in [7] does not seem to prevent it. In [21] the possibility of performing several (even an infinite number) of relativistic computational phases during a single computation is allowed. Extended relativistic mode will prove to be a highly effective intermediate. In general one should keep the number of relativistic oracle calls as low as possible, considering the physical justification. Bounding the number of oracle queries for arbitrary oracles is a well-studied issue in complexity theory (cf. [2]), which thus finds an interesting new motivation here.

2.2 Relativistic Computation

To assess the *computational power* of RTMs we assume some familiarity with the classes $\Sigma_i (i \geq 0)$ and $\Pi_i (i \geq 0)$ of the so-called *Arithmetical Hierarchy* in recursion theory, with $\Delta_i = \Sigma_i \cap \Pi_i$ for every $i \geq 0$ (cf. [14]). Recall that $\Sigma_1 =$ *'the recursively enumerable sets'* and $\Delta_1 =$ *'the recursive sets'*.

From a functional point of view the model of RTMs was considered in [7] under the name of *relativistic computer*. Using an obvious connection to classical TM-computations with the Halting Problem as oracle, it was proved in [7] that relativistic computers can decide all languages in Σ_1 and can recursively enumerate all languages in Σ_2, thus showing that relativistic computers go beyond the range of the Church-Turing thesis. The following observation can be made (modulo encodings), improving on [7].

Theorem 1 *Relativistic Turing machines (thus: relativistic computers) recognize precisely the Δ_2-sets of the Arithmetical Hierarchy.*

Proof: First we observe that extended relativistic mode can be simulated by iterating calls in normal relativistic mode, using a carefully tuned procedure in which the computation in a relativistic phase is traced from yes-answer to yes-answer (and the original instantaneous description is re-instated whenever the end of the simulation of the phase in extended mode is reached). The remainder of the argument parallels Post's theorem for Δ_2-sets (see[14]). Let A be recognized by a RTM R, with R necessarily always halting. Without loss of generality R works in normal relativistic mode. Now a recursive predicate $F(x, y, w)$ can be designed such that $w \in A \Leftrightarrow \exists_x \forall_y F(x, y, w)$, as follows. Use the $\exists_x$ to delineate a finite computation by R, extending finitely into those relativistic phases that are called and lead to a yes-signal up to the point that a yes-signal is generated, and use the $\forall_y$ to express that in this computation the relativistic phases that are called and do not lead to a yes-signal indeed do not generate such a signal in all steps to infinity. This shows that $A \in \Sigma_2$. At the same time we have $\overline{A} \in \Sigma_2$ as $\overline{A}$ is relativistically recognizable as well, hence $A \in \Pi_2$. Thus $A \in \Delta_2$.

Now let $A \in \Delta_2$. It means that there are recursive predicates F and G such that $w \in A \Leftrightarrow \exists_x \forall_y F(x, y, w)$ and $w \in \overline{A} \Leftrightarrow \exists_x \forall_y G(x, y, w)$. Now $w \in A$ can be decided by a RTM R that enumerates the possible values of $x = 0, 1, \ldots$ and checks for each value in two relativistic phases whether $\forall_y F(x, y, w)$ or $\forall_y G(x, y, w)$. For some x, hence in finite time, one of the two must signal yes. $\square$

In the model of RTMs as we described it, we abstracted from the underlying physical relativistic computational system and concentrate on the formal mechanism for invoking the *relativistic phases of computing* and for getting 'signals' from it in finite time. This makes it possible to study the effect of relativistic computations in bounded (memory-)space, very much like the complexity of oracle computations was studied.

For a RTM we define its *space complexity* $S(n)$ as usual, i.e. as the maximum number of memory-cells used on inputs of length n, taken over all inputs of length n. In this bound, the length of the rewritten part of the output tape is also included. However, *the space consumed by the spawned process in a relativistic phase of computation is not counted* in the space-complexity. Note that this space may be infinite. If only $O(1)$ extra cells are needed in addition to the input, we say that the machine operates *within constant extra space*. There seems to be no justification for defining the time complexity of a RTM.

To illustrate the power of RTMs we present two RTMs in detail that recognize specific undecidable languages. The ideas of the constructions will be used later. Let K be the set of all strings w that are descriptions of Turing machines (denoted by $\langle w \rangle$) that accept their own description. It is well-known that K is not recursive, i.e. $K \in \Sigma_1 \setminus \Pi_1$ [15].

Proposition 1 *There is a RTM R of constant extra space complexity that recognizes K. In fact, all languages of $\Sigma_1 \cup \Pi_1$ can be relativistically recognized in constant extra space and using at most one relativistic phase.*

Proof: Let R be a universal TM capable of simulating machines described on its input. R will have two signal states: the halting signal state q_H and the no-halting signal state q_N. R first reads its entire input. After reaching the end of it, it enters a relativistic state q_R in which R will start to simulate machine $\langle w \rangle$ on w. At the end of the phase, R either enters state q_H or q_N. In the former state R accepts w, whereas in the latter case it rejects it. The correctness of R follows directly from the operational semantics of RTMs after entering the signal states. Clearly R works in constant extra space. A similar argument holds for all sets in $\Sigma_1 \cup \Pi_1$. $\square$

The next proposition shows a more complicated RTM that recognizes a language by means of exponentially many relativistic sub-computations. However, far fewer phases will be seen to suffice as well. The language we consider is $K_a = \{w | MAX(w)\}$, where $MAX(w)$ holds if and only if w satisfies the following condition: $w \in K$ and, if $|w| = n$, then the running time of $\langle w \rangle$ on w is the maximum one from among all such running times of machines with description length n.

Proposition 2 *There is a RTM T of constant extra space complexity that recognizes language K_a. In fact, T needs at most two relativistic phases in normal mode, or at most one in extended mode.*

Proof: The idea is to design a RTM T that operates as follows, on any input w of length n. First it checks whether $w \in K$ and if so, it systematically generates, in lexicographic order, all descriptions v of TMs of length n. Doing so, T tries to determine those descriptions v that belong to K and satisfy $MAX(v)$. If a v is found that satisfies $w = v$ then T accepts; otherwise T rejects.

The construction in Proposition 1 shows that an RTM can easily decide whether a string of length n is a member of K. Assume that T has found two strings, u and v, from K. Now T must decide which of the two machines whose encoding is represented by u and v, has the longer running time. T can classically decide this by alternately simulating one step of each machine, to see which of the two machines halts sooner. However, in some cases this can consume an enormous amount of space which surely cannot in general be bounded by e.g. a polynomial in n. Therefore, for this particular task we design a relativistic strategy for machine T. We equip T with two halting signal states, q_1 and q_2. Entering q_1 will signal that the halting state of machine $\langle u \rangle$ is reached whereas entering q_2 will signal that the halting state of $\langle u \rangle$ is reached. Now, all that T has to do, having remembered both u and v, is to enter relativistic state q_R and spawn the alternating simulation of both machines $\langle u \rangle$ and $\langle v \rangle$ described above into the relativistic phase. When the phase ends, T can tell by the state it is in which of the two simulated machines will run longer. It is now easy to construct a RTM T that recognizes K_a, using linear space.

A RTM T that recognizes K_a in constant extra space can be obtained as follows. After determining that $w \in K$, one can relegate the entire search for (the existence of) a string $u \in K$ of length n for which $\langle u \rangle$ runs longer on u than $\langle w \rangle$ on w to a relativistic phase. Note that this would require only two relativistic phases. By setting a buffer value of 2, the process can be carried out in extended mode in one phase as follows. Carry out the computations of $\langle w \rangle$ on w and of all $\langle u \rangle$ on u simultaneously. Whenever the computation on some u ends before the computation on w, cancel the simulation on u. When the computation on w ends, emit the first yes-signal and proceed with the remaining simulations. The first one that now halts will emit a yes-signal as well, but no further yes-signals are emitted after this. (By the bound on the buffer the relativistic phase ends in fact when a second yes-signal is reached.) T can now decide whether $w \in K_a$ by just looking at the buffer value when it returns from the relativistic phase. $\square$

2.3 Non-uniform computations

We will compare RTMs with *Turing machines with advice* (TM/A's) (cf. [1, 9, 12]), to establish the link with non-uniform complexity theory. A TM/A is a classical Turing machine enhanced by an *advice function.* An advice function is a function $f : \mathbf{Z}^+ \to \Sigma^*$. For given n, $f(n)$ is called the advice for length n. An advice is called $S(n)$-bounded if for all n, the length of $f(n)$ is bounded by $S(n)$.

A TM/A operating on an input of size n, is allowed to call the value of its advice function only once during the computation, and it can call it only for the particular n. To realize an advice-call, a TM/A is equipped with a separate *advice tape* (that is initially empty) and a distinguished *advice state.* By entering the advice state at time t the value of $f(n)$ will appear on the advice tape in a single step at time $(t + 1)$. The mechanism of advice is very powerful and can provide a TM/A with highly non-recursive 'assistance'. Without further constraints, advice functions are as powerful as arbitrary oracles. We will be interested in advice functions whose values are bounded in length by some (computable) functions of n. Let $\mathsf{TM}/poly$ be the class of languages recognized by TM/A's with polynomially-bounded advice (cf. [1, 9, 12]). To prepare for a later construction, we show that $K \in \mathsf{TM}/poly$ (cf. Proposition 1).

Proposition 3 *There is a TM/A A with a linearly bounded advice function f that recognizes the language K and halts on all inputs.*

Proof: Define the advice function f as follows: $f(n) = u$ if and only if $|u| = n$, and u is the lexicographically first string that satisfies $MAX(u)$ (the predicate MAX was defined before the statement of Proposition 2). A default value should be used in case no u with $|u| = n$ of the required property exists. Clearly, f is linearly bounded. On input w of length n, A works as follows. First, it checks whether w is the encoding of some Turing machine M. If so, then A calls its advice to get $f(n) = u$. Then A makes use of the idea we already saw in the proof of Proposition 2: A starts to alternate between simulating one step of M on u, and one step of $\langle w \rangle$ on w. Clearly, the simulation of $\langle u \rangle$ must terminate, and if it terminates earlier than the simulation of M then, thanks to the truth of $MAX(u)$, A can infer that M on w will never stop. Then A rejects w and halts. Otherwise, if the simulation of $\langle w \rangle$ halts sooner than the simulation of $\langle u \rangle$, A accepts w and halts. $\square$

Note that all three previous propositions are related: we have a (non-recursive) language K recognized both by a RTM (Proposition 1) and by a TM/A (Proposition 3). Moreover, the advice function used by the TM/A is computable by a RTM (Proposition 2). In the next section we show that this is not a coincidence.

3 Relativistic computing vs. non-uniform computing

Now we can formulate the main result, relating infinite relativistic and non-uniform finite computations. Let P be an arbitrary observable, i.e. recursive property (e.g. 'halting'), and let $\mathsf{RelSPACE}(P, S)$ be the family of $S(n)-$bounded advice functions that are $S(n)-$space bounded computable by an RTM on input of any length$-n$ string using the relativistic oracle for P (in normal mode). We will prove that space-bounded RTMs, i.e. Turing machines with relativistic oracles, are equivalent to Turing machines with advice, assuming the latter take their advices from a suitable family. Assume that $S(n)$ is space-constructible, and let $\sharp$ be a marker symbol.

Lemma 1 *Let language L be recognized by a $S(n)-$space bounded RTM. Then there is a function $f \in \mathsf{RelSPACE}(P, S)$ such that $L' = \{w\sharp f(|w|) | w \in L\}$ is recognizable by a linear-space bounded RTM that needs at most two relativistic phases on every input.*

Proof: Let L be recognized by some $S(n)-$space bounded RTM R. First we note that R can be assumed to always halt on every input, by the classical fact that one could prevent it from cycling using a $S(n)-$space counter otherwise. To construct f, consider how R acts on inputs w with $|w| = n$ and try to eliminate the need for 'infinite time' relativistic phases. The idea is to let f provide an upper bound on the length of all relativistic phases in R's computations on inputs of length n that eventually lead to a yes-signal and thus end by R's entering into the corresponding yes-signal state. We do this by letting $f(n) = 0x_n$, where x_n is an instantaneous description of R that triggers the longest relativistic phase of this kind of all such ID's of length $\leq S(n)$. To let this be well-defined, we let x_n be the lexicographically smallest ID with this property and take $f(n) = \epsilon$ if no ID of this kind exists. If R could get a purported value x of $f(|w|)$ from somewhere on input w, it might act as follows: verify that $x = f(|w|)$ in one relativistic phase, and run a classical simulation of R in another, replacing every call to the relativistic oracle by a direct computation dovetailed with the simulation of R on x while continuously checking P (and drawing the obvious conclusion if no yes-signal has been reached by the time the run on x ends).

Following this idea, we design a RTM R' to recognize L' as follows. On receiving its input, R' checks whether it is of the form $w\sharp x$. It then proceeds to check that $x = f(|w|)$ as follows. The strategy is similar to the strategy used in Proposition 2. Note that the computation in any relativistic phase is fully determined by the ID of R at the beginning of this phase, and every ID that triggers a relativistic phase will be bounded in length by $S(n)$, with $n = |w|$. R' first checks whether x triggers a relativistic phase of R that leads to a yes-signal, by actually performing the relativistic phase if it is triggered ('phase 1'). If x passes this test, R' starts up a second relativistic phase ('phase 2') in which it constructs (the value of) $S(n)$, enumerates all ID's $y \neq x$ of length $\leq S(n)$ that R could admit on any input of length n (and that trigger a relativistic phase), starts a simulation of R on x and on all y's, and alternately carries out one step in each of the simulations. Whenever a computation on an ID $y \neq x$ ends (i.e. reaches a yes-signal) while the computation on x is still going on, the simulation on y ends and is removed (and the yes-signal is *not* sent). If no simulation on any $y \neq x$ survives before the simulation on x reaches its yes-signal, the simulation process is sent into an infinite loop (without generating a yes-signal ever). If some simulations on ID's $y \neq x$ have survived by the time the simulation on x reaches its yes-signal, the simulation on x ends and is removed (and the yes-signal is *not* sent) and the simulation on the remaining ID's $y \neq x$ is continued. Whenever one of these remaining simulations reaches a yes-signal, this yes-signal is actually sent. Clearly the outcome of this relativistic phase will tell R' whether x is an ID that triggers the longest relativistic phase that leads to a yes-signal. The given argument is easily modified to include the check that x actually is the lexicographically smallest with this property. So far no more than linear space was used (i.e. linear in $|w\sharp x|$).

Assuming $x = f(|w|)$, R' can now begin the actual recognition process of the string w. In principle R' could avoid all relativistic phases in the simulation of R, because it can bound how far it must go into any such phase in a classical simulation before knowing whether a yes-signal will be reached or not: R' can simply interleave the simulation with the simulation of R on x and use the latter as a yardstick. If no yes-signal is reached before the end of the yardstick, it knows that a yes-signal will never be reached and that it can proceed to q_N. To avoid the space-count for this, R' relegates the recognition process to a relativistic phase ('phase 3'): in this phase it simulates R on w, using (the computation on) x as the yardstick to do a finite classical simulation of every relativistic phase which R would otherwise have to carry out, and emitting a yes-signal if and when the simulation thus carried out actually lead to the recognition that $w \in L$. Observe that phase 3 always halts, because R is always halting. This implies that R' can combine phases 1 and 3 into one: a transfer to q_N will now mean that either $x \notin K$, or $x \in K$ and either $w \notin L$ or the computation wasn't long enough to reach the conclusion that $w \in L$, i.e. $x \neq f(|w|)$. Thus, altogether R' needs only two relativistic phases.

Finally, observe that the function f is computable by an RTM in space $S(n)$ on any input of length n, by very much the same argument as employed above. A RTM R'' for it will first construct $S(n)$ space and then use it to enumerate every candidate ID y explicitly, testing for each y whether it qualifies (like we did for x in phase 1) and running a contest between y's that qualify as in the proof of Proposition 2. This can easily be carried out within space $S(n)$. Once the winning string x_n has been found (or no string was found to qualify), the value of $f(n)$ can be output. $\square$

By a further argument one can show that the language L' can be recognized using at most *one* relativistic phase in extended mode (but using $S(n)$ space).

We can now prove the main result. In the result we assume that the Turing machines with advice can check property P effectively, either as a subroutine or as an oracle. Recall that functions in the set $\mathsf{RelSPACE}(P, S)$ are always $S(n)-$bounded.

Theorem 2 *The following are equivalent, for any language L:*

(i) L is recognized by a $S(n)-$space bounded RTM.

(ii) L is recognized by a TM/A that uses an advice function $\in \mathsf{RelSPACE}(P, S)$.

Proof:

$(i) \rightarrow (ii)$ Let L be recognized by a $S(n)-$space bounded RTM R. By Lemma 1 there exists a function $f \in \mathsf{RelSPACE}(P, S)$ such that $L' = \{w\sharp f(|w|) | w \in L\}$ is recognizable by a (linear-space) RTM R' that uses at most three relativistic phases. Consider the particular RTM R' that was constructed in Lemma 1 and design a Turing machine T with advice f as follows. On input w of length n, T immediately calls its advice and lays out the string $w\sharp f(|w|)$. T now prepares to simulate R'. Clearly it can skip phases 1 and 2 of R' and move straight to phase 3. T now carries out the simulation in phase 3 in an entirely classical way just like it was specified for R', using the yardstick idea to 'finitize' the relativistic phases of R. T accepts if and only if the computation leads a yes-signal.

$(ii) \rightarrow (i)$ Let L be recognized by a TM/A T using an advice $f \in \mathsf{RelSPACE}(P, S)$. Design a RTM R for L as follows. On input w, R first computes the string $f(n) = f(|w|)$ in space $S(n)$. It then triggers a relativistic phase in which it does the entire simulation of T, having the advice-value available whenever T calls for it. The relativistic phase triggers a yes-signal when T accepts. When R observes the result and enters into a yes-state, it accepts w. Clearly R recognizes L and does so in space S(n). $\square$

In the theorem we can assume without loss of generality that the machines involved in (i) and (ii) halt on all inputs. Specializing the result to the case considered in [7], let P = '*halting*' and let $\mathsf{RelPSPACE} = \bigcup_k \mathsf{RelSPACE}(P, n^k)$. Let $\mathsf{RTM\text{-}PSPACE}$ denote the class of languages recognizable by RTMs in polynomial space.

Corollary 1 *The following are equivalent, for any language L:*

(i) L is recognized by a polynomial-space bounded RTM.

(ii) L is recognized by a TM/A that uses an advice function $\in \mathsf{RelPSPACE}$.

Consequently, $\mathsf{RTM\text{-}PSPACE} \subseteq \mathsf{TM}/poly$.

Using well-known facts from non-uniform computational complexity (cf. [1])it follows that computations by RTMs are as powerful as computations of infinite families of (finite) non-uniform circuits whose size grows faster than any recursive function. A further observation can be made that shows the 'physical' efficiency of relativistic phases in extended mode.

Theorem 3 *If a language L is recognized by a $S(n)-$space bounded RTM, then it can be recognized by a $S(n)-$space bounded RTM that needs at most two relativistic phases, one in extended mode and one in normal mode.*

Proof: Let L be recognized by a $S(n)-$space bounded RTM R. By Theorem 2, L can be recognized by a TM/A T, using the advice function $f \in \mathsf{RelSPACE}(P, S)$ constructed in the proof of Lemma 1. But a RTM can compute $f(n)$ as follows. Using one relativistic phase in extended mode, the RTM can count the number of ID's y of length $\leq S(n)$ that trigger a relativistic phase that actually leads to a yes-signal ('halts'). Once it knows this number, R can determine in a completely classical dovetail which of the computations on these ID's actually halt and what the lexicographically smallest y is that leads to the longest halting computation among them. (This 'census approach' is well-known in complexity theory, cf. [9].) This shows that L can be recognized by a RTM R that computes like T and computes the advice value when it is needed, using at most one relativistic phase in extended mode for this purpose and no other relativistic calls. In order to achieve the bounded space complexity, let R relegate the entire (classical) simulation of T to a relativistic phase in normal mode. In this way R remains relativistically $S(n)-$space bounded and recognizes L by means of at most two relativistic phases. □

4 Conclusion

Using arguments from general relativity theory, Etesi and Németi [7] have argued that the Church-Turing thesis may be physically untenable. They have shown that relativistic computers can be thought of that can recognize non-recursive languages. We have formalized the model of relativistic computing and characterized its potential precisely from the viewpoint of computability theory. We also proved a bridging result that links the infinite computations of (space-bounded) relativistic TMs to finite computations by TMs with powerful advice functions.

The result, proving a kind of duality between infinite relativistic and non-uniform finite computations, follows from basic complexity-theoretic considerations and should be of interest for the appraisal of relativistic computing in the context of (theoretical) physics or philosophy. The result can be seen as a further evidence for the emerging central role of non-uniform computation models in capturing the information processing capabilities in natural systems, as formulated in [17]. The result complements the existing set of examples of such kinds of models, including evolutionary interactive systems, the Internet, artificial living systems, social systems, and amorphous computing systems (cf. [18]), by systems operating by the principles of general relativity theory. Undoubtedly the result, suggesting a kind of duality between infinite and non-uniform finite computations, is also interesting in the context of (theoretical) physics or philosophy.

Bibliography

[1] J. L. Balcázar, J. Díaz, J. Gabarró: *Structural complexity I*, Second Edition, Springer-Verlag, Berlin, 1995.

[2] R. Beigel, *Query-limited reducibilities*, PhD thesis, Department of Computer Science, Stanford University, Stanford, 1995, available at www.eccc.uni-trier.de/eccc-local/ECCC-Theses/beigel.html.

[3] M. Blum, F. Cucker, M. Shub, M. Smale: *Complexity and real computation.* Springer, New York, 1997, 453 p.

[4] M. Burgin: How we know what technology can do, *Communications of the ACM* 44 (2001) 83-88.

[5] C.S. Calude, Pavlov, B.: Coins, quantum measurements, and Turing's barrier, Technical Report CDMTCS-170, University of Auckland, New Zealand, December 2001.

[6] J. Earman, J. Norton: Infinite pains: The trouble with supertasks, in A.Morton and S.P. Stich (eds.), *P. Benacerraf and his critics*, Philosophers and Their Critics Vol 8, Blackwell Publ., Oxford (UK) / Cambridge (MA), 1996, Ch 11, pp 231-261.

[7] G. Etesi, I. Németi: Non-Turing computations via Malament-Hogarth space-times, *Int. J. Theor. Phys.* 41 (2002) 341-370, see also: http://arXiv.org/abs/gr-qc/0104023.

[8] J.D. Hamkins, A. Lewis: Infinite time Turing machines, *Journal of Symbolic Logic* Vol. 65, No. 2, pp. 567-6-4, 2000.

[9] L.A. Hemaspaandra, M. Ogihara: *The complexity theory companion*, Springer-Verlag, Berlin, 2002.

[10] J. Hogarth: Non-Turing computers and non-Turing computability, in D. Hull, M. Forbes and R.M. Burian (eds.), *1994 Biennial Meeting of the Philosophy of Science Association* (PSA 1994), Proceedings Vol. One, Philosophy of Science Association, East Lansing, 1994, pp 126-138.

[11] J. Hopcroft, R. Motwani, and J.D. Ullman: *Introduction to automata theory, languages and computation*, 2nd Edition, Addison-Wesley, Reading, MA, 2000.

[12] R.M. Karp, R.J. Lipton: Some connections between non-uniform and uniform complexity classes, in *Proc. 12th Annual ACM Symposium on the Theory of Computing* (STOC'80), 1980, pp. 302-309.

[13] J.D. Norton: What can we learn about ontology of space and time from the theory of relativity?, *PhilSci Archive* (http://philsci-archive.pitt.edu/), ID: PITT-PHIL-SCI00000138, 2001.

[14] H. Rogers Jr, *Theory of recursive functions and effective computability*, McGraw-Hill, New York, 1967.

[15] A.M. Turing: On computable numbers, with an application to the Entscheidungsproblem, *Proc. London Math. Soc.,* 42-2 (1936) 230-265; A correction, *ibid*, 43 (1937), pp. 544-546.

[16] A.M. Turing: Systems of logic based on ordinals, *Proc. London Math. Soc. Series* 2, 45 (1939), pp. 161-228.

[17] J. van Leeuwen, J. Wiedermann: The Turing machine paradigm in contemporary computing, in: B. Enquist and W. Schmidt (Eds.), *Mathematics unlimited - 2001 and beyond*, Springer-Verlag, 2001, pp. 1139-1155.

[18] J. van Leeuwen, J. Wiedermann: Beyond the Turing limit: evolving interactive systems, in: L. Pacholski, P. Ružička (Eds.), *SOFSEM'01: Theory and Practice of Informatics*, 28th Conference on Current Trends in Theory and Practice of Informatics, Lecture Notes in Computer Science Vol. 2234, Springer-Verlag, Berlin, 2001, pp. 90–109.

[19] P. Wegner: Why interaction is more powerful than algorithms, *C. ACM* 40, (1997) 315-351.

[20] J. Wiedermann, J. van Leeuwen: Emergence of super-Turing computing power in artificial living systems, in: J. Kelemen, P. Sosík (Eds.), *Artificial Life 2001*, 6-th European Conference (ECAL 2001), Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, 2001, pp. 55-65.

[21] L. Wischik: *A formalisation of non-finite Computation*, dissertation submitted to the University of Cambridge towards the degree of Master of Philosophy, June 1997 (available at: www.wischik.com/lu/philosophy/non-finite-computation.html).