Space bounds for infinitary computation

Benedikt Löwe*

Institute for Logic, Language and Computation, Universiteit van Amsterdam,
Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands

Mathematisches Institut, Rheinische Friedrich-Wilhelms-Universität Bonn,
Beringstraße 1, 53115 Bonn, Germany

Fachbereich Mathematik, Universität Hamburg, Bundesstrasse 55, 20146 Hamburg,
Germany

e-mail: `bloewe@science.uva.nl`

Infinite Time Turing Machines (or Hamkins-Kidder machines) have been introduced in [HaLe00] and their computability theory has been investigated in comparison to the usual computability theory in a sequence of papers by Hamkins, Lewis, Welch and Seabold: [HaLe00], [We00a], [We00b], [HaSe01], [HaLe02], [We04], [We05] (*cf.* also the survey papers [Ha02], [Ha04] and [Ha05]). Infinite Time Turing Machines have the same hardware as ordinary Turing Machines, and almost the same software. However, an Infinite Time Turing Machine can continue its computation if it still hasn't reached the Halt state after infinitely many steps (for details, see § 1).

In [Sc03], Schindler started the investigation of the corresponding complexity theory by defining natural time complexity classes for Infinite Time Turing Machines. Schindler, Welch, Hamkins and Deolalikar have proved with methods of descriptive set theory that the big open questions of standard complexity theory $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ and $\mathbf{P} \stackrel{?}{=} \mathbf{NP} \cap \mathbf{coNP}$ have negative answers for Infinite Time Turing Machines [Sc03,DeHaSc05,HaWe03].

For an ordinary Turing machine that stops in a finite number t of steps, it is easy to define its space usage: during its computation, it has used at most t cells of the tape, possibly less. This finite number of used cells can serve as a measure of space usage. A halting computation will have used a finite amount of time and space; if, however, time or space usage are infinite, then this corresponds to usage of order type ω and automatically implies that the computation was non-halting.

In this paper, we shall consider both Hamkins-Kidder machines and Koepke's *Ordinal Machines* as described in [$\text{Ko}_0$05a] and [$\text{Ko}_0$05b]. Koepke machines can not only extend their computation into transfinite ordinal time, but they also have ordinal-indexed cells on their tapes. Therefore, there is a natural notion of space usage for computations on Koepke machines that corresponds to the classical idea of space constraints on Turing Machines: just count the number (order type) of cells being used.

* The author thanks Joel Hamkins (New York NY), Peter Koepke (Bonn), Philip Welch (Bristol) and Joost Winter (Amsterdam) for discussions about infinitary computation, and the anonymous referees for important comments.

This is very different for Hamkins-Kidder machines whose space is constrained to a tape of order type ω whereas time can have arbitrary ordinals as order type. This asymmetry makes is hard to give a definition of space usage that can be compared to time usage.

In this paper, we discuss the basics of possible definitions for space constraints for the mentioned two types of infinitary Turing machine computations. In § 1, we give all definitions needed in the paper and then briefly discuss space constraints for Koepke's machines in § 2 and space constraints for Hamkins-Kidder machines in § 3. Finally, in § 4, we discuss nondeterministic computation.

This paper raises very general questions about infinitary algorithms. We list them here and will explain the questions in more detail in the respective sections:

1. Are there any algorithms for Koepke's ordinal machines that use the additional transfinite space in order to compute more than Hamkins-Kidder machines within time restrictions? (§ 2; Theorem 3 gives an example of a use of the additional space, but it is not time efficient.)
2. Are there any algorithms for Hamkins-Kidder machines that compute complicated sets with unlimited time but very simple snapshots on the scratch tape? (§ 3; Question 10 provides a very basic test question.)
3. Are there any nondeterministic algorithms that are space efficient? (§ 4; Proposition 11 gives a general description of nondeterministic algorithms that mimic *guess nondeterminism*, but they are not space efficient.)

1 Definitions

In the following, we shall give a description of both Hamkins-Kidder machines and Koepke's ordinal machines.

Like ordinary Turing machines, both types of infinitary Turing machines consist of an **input tape**, a **scratch tape** and an **output tape**, a **reading/writing head**, a finite set of **states** and a program δ that assigns to a state s and the content of a bit on the scratch tape and the input tape an **action**. The action consists of moving the head right, moving the head left, writing on the output tape, writing on the scratch tape, erasing on the scratch tape, or a combination of these actions. Note that we may not erase on the output tape; this is to make sure that the program doesn't abuse the output tape as a scratch tape (in our definition of the space complexity, we shall only count the complexity of snapshots on the scratch tape).[1]

In the case of the Hamkins-Kidder machines, all of the three tapes have order type ω (as for ordinary Turing machines), in the case of the Koepke machines, the tapes are class-sized with a cell for every ordinal. If we have a class-sized tape, then we have to say what the machine will do if it in a cell indexed with a limit ordinal and receives the comment "move left". In that case, we'll move the reading/writing head to the 0th cell.

[1] Since we're only discussing decision problems here, *i.e.*, the output is either 0 or 1, this is equivalent to saying that the output tape has only one bit.

If we fix a machine T, an input x and an appropriate time α, then we write $s^T_\alpha(x)$ for the state that the machine is in and $h^T_\alpha(x)$ for the position of the head at time α with initial input x. If β is an index for a cell on the scratch tape, then we write $c^T_\alpha(x,\beta)$ for its content at time α. We also use $c^T_\alpha(x)$ for the function $\beta \mapsto c^T_\alpha(x,\beta)$ which we call the **snapshot at time** α. Note that this a function with domain ω for Hamkins-Kidder machines, and a class function with domain Ord for Koepke machines.

For finitary computation, the times α mentioned in the last definitions are always finite. Infinitary Turing machines differ from a normal Turing machines in that they are allowed to continue their computation beyond ω many stages of computation. At a limit step of the computation, all the cells on the tape are adjusted according to the limit behaviour of the entries along the infinite computation: If 0 occurred cofinally often, the cell will get value 0 in the limit step, if on the other hand, from a point on, 1 was written in the cell, the cell will get value 1 (this corresponds to taking the *liminf* of the cell values). The state of the machine at a limit stage λ will also be the *liminf* of the states below λ. Note that while this is the definition from [Ko$_0$05a] for Koepke machines, it is not the standard definition for Hamkins-Kidder machines: in [HaLe00], these have a designated LIMIT state that is assumed in all limit stages. In terms of computational power, the two definitions for Hamkins-Kidder machines don't make a difference (as long as we have more than one tape).

The position of the reading/writing head at a limit stage λ is where our two infinitary models differ: For the Hamkins-Kidder machines, the head will always be moved to cell 0 at a limit stage: consequently, the head will never move to a cell indexed with an infinite ordinal. For the Koepke machines, the head will be moved to

$$h^T_\lambda(x) := \liminf_{s^T_\alpha(x)=s^T_\lambda(x)} h^T_\alpha(x).$$

Let us summarize the behaviour of the three types of machines in the following table:

	time	tape(s)	cells at limit	head at limit
Turing machines	ω	ω	N.A.	N.A.
Hamkins-Kidder machines	Ord	ω	lim inf	first cell
Koepke machines	Ord	Ord	lim inf	lim inf

We say a machine **accepts** an input x if it reaches the HALT state and has 1 on the output tape at that time. If it yields 0, we say that it **rejects** the input x. A set A is **(Turing, Hamkins-Kidder, Koepke) decidable** if there is a (Turing, Hamkins-Kidder, Koepke) machine that accepts exactly the elements of A and rejects exactly the reals not in A. Let us denote the set of Turing (Hamkins-Kidder, Koepke) decidable sets by $\mathbf{Dec}^{\mathrm{T}}$ ($\mathbf{Dec}^{\mathrm{HK}}$, $\mathbf{Dec}^{\mathrm{K}}$).

For all of the three types of machines, there is an obvious definition of time usage for a halting computation: if T is a machine of the appropriate type that reaches the HALT state at input x, then we write $\mathbf{time}(x,T)$ for the first α such that $s^T_\alpha(x) = \text{HALT}$. If $f : \mathbb{R} \to \text{Ord}$ is a function assigning ordinals to inputs, we say that T is an **time f machine** (or more simply, an f-machine) if for all x, we have $\mathbf{time}(x, T) < f(x)$.

Following Schindler [Sc03], we write $\mathbf{P}_f$ for the class of all sets of reals that are decidable by f-machines. If f is the constant function $f(x) = \xi$, we also call f-machines ξ-machines. We write $\mathbf{P}_\xi$ for the class of all sets decidable by an η-machine for some $\eta < \xi$. In order to distinguish the time classes for our types of machines, we write $\mathbf{P}_f^{\mathrm{HK}}$ and $\mathbf{P}_\xi^{\mathrm{HK}}$ for the time classes for Hamkins-Kidder machines and $\mathbf{P}_f^{\mathrm{K}}$ and $\mathbf{P}_\xi^{\mathrm{K}}$ for the time classes for Koepke machines.

Note that for Turing machines and Hamkins-Kidder machines, there is only a set of snapshots, whereas for Koepke machines, there is a proper class of snapshots. This simple observation has a portentous consequence for Hamkins-Kidder machines: they have far more time at their disposal than there are possible computation situations. If there are two limit ordinals $\lambda < \lambda^*$ such that the computation at λ and λ^* has the same state and snapshot and none of the cells with the value 1 at λ changes its value between λ and λ^*, we call this a **looping situation**.

Observation 1 (Hamkins-Lewis) *A Hamkins-Kidder machine does not halt if and only if its computation has a looping situation.*

Proof. [HaLe00, Corollary 1.2].[2] q.e.d.

The analogue of Observation 1 is not true for Turing and Koepke machines, as they have exactly as much time as there are snapshots. In both cases consider the empty input and the machine that writes 1 and moves right if it hits a 0. This machine will never halt nor loop.

Another observation that will be important is that infinitary computations can be done in initial segments of the constructible hierarchy. For a Hamkins-Kidder machine T and any time α and input x, let $c_\alpha^T(x)$ be the content of the full tape (of order type ω) at time α (with input x and machine T).

Observation 2 *For any Hamkins-Kidder machine T, any ordinals $\alpha < \xi$ such that ξ is admissible, and any $x \in 2^\omega$, we have that*

$$c_\alpha^T(x) \in \mathbf{L}_\xi[x].$$

2 Koepke's ordinal machines

Koepke's analysis of ordinal machines from [Ko$_0$05a,Ko$_0$05b,Ko$_0$Ko$_1\infty$] does not pay attention to either computing resources or real numbers. Whereas we are interested in decision problems, he is interested in creating (characteristic

[2] The diligent reader checking this again [HaLe00] might notice that they write “the cells which are 0 at the limit never subsequently turn into 1 (we allow the 1s to turn to 0 and back again)”. This is due to the fact that [HaLe00] uses a *limsup* rule instead of our *liminf* rule.

functions of) sets on the output tape and allows as input finite sets of ordinals as parameters.

However, if you restrict your attention to decision problems, Koepke machines are still more expressive than Hamkins-Kidder machines as the following result shows:

Theorem 3 *The halting problem for Hamkins-Kidder machines is Koepke decidable. Hence, the set of Koepke decidable sets of reals is strictly bigger than that of Hamkins-Kidder decidable sets.*

Proof. By [HaLe00, Theorem 4.1], the Hamkins-Kidder halting problem is not Hamkins-Kidder decidable but semi-decidable. We only have to give an algorithm to decide the complement of the Hamkins-Kidder halting problem with a Koepke machine.

We shall be using Observation 1. In order to find out whether a computation doesn't halt, we can just check whether a looping situation occurred in the computation.

Since a Koepke machine has an unlimited amount of space, and every Hamkins-Kidder situation can be coded as a real, we can simulate the run of a Hamkins-Kidder machine while keeping track of the entire computation on the class-sized tape. It is now easy to check whether a looping situation occurred. q.e.d.

If you look at the algorithm used to decide the Hamkins-Kidder halting problem in this proof, you'll notice that it is neither time nor space efficient. This is a general problem with complexity theory for Koepke machines: while Theorem 3 proves that you can use the size of the tape to compute more, there is no known technique to use it in order to compute faster.

As a consequence, we do not know any non-trivial separation results of time complexity for Hamkins-Kidder machines and Koepke machines.

Proposition 4 *If* $\omega^2 \leq \alpha \leq \omega_1^{\mathrm{CK}}$*, then* $\mathbf{P}^{\mathrm{K}}_\alpha = \mathbf{P}^{\mathrm{HK}}_\alpha$.

Proof. Fix $\eta < \alpha$. Given a set A that is Koepke decidable by an η-machine, we will describe how we decide it with a Hamkins-Kidder η-machine. Note that the original η-machine can never use more than the first η many cells of the class-size tape.

Since $\eta < \omega_1^{\mathrm{CK}}$, we can produce a code of η on the scratch tape within ω steps. After that, we use that code in order to think of the ω-tape as an η-tape and run the Koepke algorithm on it. This combined algorithm takes $\omega + \eta = \eta$ steps. q.e.d.

What if we allow a Koepke machine more than a constant amount of time? Let $f_0(x) := \omega_1^x$. Then what is $\mathbf{P}^{\mathrm{K}}_{f_0}$? By [DeHaSc05, Theorem 4.2 (ii)], $\mathbf{P}^{\mathrm{HK}}_{f_0} = \mathbf{P}^{\mathrm{HK}}_{\omega_1^{\mathrm{CK}}}$; this was strengthened by Welch [We06, Proposition 2] to the following result:

Proposition 5 (Welch) *Every* f_0*-machine is an* ω_1^{CK}*-machine.*

An analogue of Proposition 5 for Koepke machines together with Proposition 4 would show that $\mathbf{P}^{\mathrm{K}}_{f_0} = \mathbf{P}^{\mathrm{HK}}_{f_0}$.

For a Koepke machine T, an input x and a time α, we write $u^T_\alpha(x) := \sup\{\beta\,;\,c^T_\alpha(x,\beta) \neq 0\}$ for the space used at time α. If a Koepke machine with input x reaches the HALT state at time $\mathbf{time}(x,T)$, we write

$$\mathbf{space}(x,T) := \sup\{u^T_\xi(x)\,;\,\xi < \mathbf{time}(x,T)\}.$$

If $f : \mathbb{R} \to \mathrm{Ord}$ is a function assigning ordinals to inputs, we say that T is an **space** f **machine** if for all x, we have $\mathbf{space}(x,T) < f(x)$. In analogy to Schindler's **P**-notation for time complexity, we write $\mathbf{PSPACE}^{\mathrm{K}}_f$ for the class of all sets decidable by space f machines and $\mathbf{PSPACE}^{\mathrm{K}}_\xi$ for the class of all sets decidable by space η machines for some $\eta < \xi$.

As for ordinary Turing machines, we immediately get $\mathbf{P}^{\mathrm{K}}_f \subseteq \mathbf{PSPACE}^{\mathrm{K}}_f$ (for all functions f), as each new used cell on the scratch tape requires one unit of time to be used or skipped.

Since Hamkins-Kidder machines are essentially just space ω Koepke machines, we can mimic arbitrary Hamkins-Kidder computations with space-bounded Koepke machines:

Proposition 6 $\mathbf{Dec}^{\mathrm{HK}} \subseteq \mathbf{PSPACE}^{\mathrm{K}}_{\omega+2}$.

Proof. Let T be a Hamkins-Kidder machine deciding A. Based on T, we construct a Koepke machine that can recognize when it is in a limit stage and that, whenever it is in a limit stage, moves to cell 0. (If it happens to be in cell ω, moving one step left will move the head to cell 0.) This machine mimics the limit behaviour of T and uses the cells up to the ωth cell, *i.e.*, is a space $\omega + 1$ machine. q.e.d.

Corollary 7 $\mathbf{P} \neq \mathbf{PSPACE}$ *for Koepke machines, i.e.,* $\mathbf{P}^{\mathrm{K}}_{\omega^\omega} \neq \mathbf{PSPACE}^{\mathrm{K}}_{\omega^\omega}$.

Proof. Propositions 4 and 6 yield this simple separation result as follows:

$$\mathbf{P}^{\mathrm{K}}_{\omega^\omega} = \mathbf{P}^{\mathrm{HK}}_{\omega^\omega} \subsetneqq \mathbf{Dec}^{\mathrm{HK}} \subseteq \mathbf{PSPACE}^{\mathrm{K}}_{\omega+2} \subseteq \mathbf{PSPACE}^{\mathrm{K}}_{\omega^\omega}.$$

q.e.d.savit

3 Hamkins-Kidder machines

For Hamkins-Kidder machines, the function $u^T_\alpha(x) := \sup\{\beta\,;\,c^T_\alpha(x,\beta) \neq 0\}$ as defined above will be equal to ω as soon as the entire tape is being used, and consequently for almost all non-trivial T and x, we'll have $\mathbf{space}(x,T) = \omega$. Thus, we have to use a different approach in order to get an informative measure for space usage.

In this section, we shall give two different definitions of space usage for Hamkins-Kidder machines. As above, let $c^T_\alpha(x)$ be the content of the full tape (of order type ω) at time α (with input x and machine T). We can define

$$\ell^T_\alpha(x) := \min\{\eta\,;\, c^T_\alpha(x) \in \mathbf{L}_\eta[x]\}, \text{ and}$$

$$\mathbf{space}^0(x,T) := \sup\{\ell^T_\xi(x)\,;\, \xi < \mathbf{time}(x,T)\}.$$

As before, we define a notion of $\mathbf{space}^0$ f **machine** for Hamkins-Kidder machines and derive a definition of the class $\mathbf{PSPACE}^{\mathrm{HK},0}_f$ from it. Let us call a function $f : \mathbb{R} \to \mathrm{Ord}$ **admissible** if for all x, the ordinal $f(x)$ is x-admissible.

Proposition 8 *For any admissible function f, we have* $\mathbf{P}^{\mathrm{HK}}_f \subseteq \mathbf{PSPACE}^{\mathrm{HK},0}_f$.

Proof. Let T be a machine deciding A in time f. That means that for all inputs x, the computation has length shorter than $f(x)$. Fix some $\xi < f(x)$. By Observation 2, we have that $c^T_\xi(x) \in \mathbf{L}_{f(x)}[x]$. q.e.d.

Alternatively, we define

$$\mathbf{space}^1(x,T) := \sup\left\{\omega_1^{c^T_\xi(x)}\,;\, \xi < \mathbf{time}(x,T)\right\} + 1,$$

and a notion of $\mathbf{space}^1$ f **machine** and the class $\mathbf{PSPACE}^{\mathrm{HK},1}_f$.

Proposition 9 *For any admissible function f, we have* $\mathbf{P}^{\mathrm{HK}}_f \subseteq \mathbf{PSPACE}^{\mathrm{HK},1}_f$.

Proof. Let T be a machine deciding A in time f. Fix x, then the computation with input x has length shorter than $f(x)$. As in the proof of Proposition 8, we know that $\{c^T_\xi(x)\,;\, \xi < \mathbf{time}(x,T)\} \subseteq \mathbf{L}_{f(x)}[x]$.

Towards a contradiction, assume that $\mathbf{space}^1(x,T) \geq f(x)$, so there must be some ξ such that $\omega_1^{c^T_\xi(x)} = f(x)$. But then there is a code z for $f(x)$ that is (Turing)-recursive in $c^T_\xi(x) \in \mathbf{L}_{f(x)}[x]$, and hence $z \in \mathbf{L}_{f(x)}[x]$, contradicting the x-admissibility of $f(x)$. q.e.d.

Propositions 8 and 9 are instances of the slogan "Using space costs time". This is equally true for classical, finitary complexity theory. Even for finite time Turing machines, it is not known whether $\mathbf{P} \subsetneq \mathbf{PSPACE}$. This question can be rephrased as

> "Are there space-efficient algorithms for problems that cannot be solved quickly?"

Of course, this question can be applied to the three relations

$$\mathbf{P}^{\mathrm{K}}_f \subseteq \mathbf{PSPACE}^{\mathrm{K}}_f,$$
$$\mathbf{P}^{\mathrm{HK}}_f \subseteq \mathbf{PSPACE}^{\mathrm{HK},0}_f, \text{ and}$$
$$\mathbf{P}^{\mathrm{HK}}_f \subseteq \mathbf{PSPACE}^{\mathrm{HK},1}_f$$

as well. The first algorithm that comes to mind is the Hamkins-Lewis algorithm for deciding Π^1_1 sets [HaLe00, Count-Through Theorem 2.2]: it is not in $\mathbf{P}_{f_0}$ for $f_0 : x \mapsto \omega_1^x$; however, it is easily seen that this algorithm produces the ill-founded part of the relation coded by x on the scratch tape which is not in general in $\mathbf{L}_{f(x)}[x]$. As a consequence, the algorithm uses both a lot of time and a lot of space, and is no answer to the above question.

This example is illustrative in the following sense: looking at the different infinitary algorithms that are at our disposal, the only way that they use their infinite time is to produce more complicated reals on the scratch tape. This observation might lead to the conjecture that time and space complexity for infinitary computation are the same.

Let us highlight this bold conjecture with a precise test question: Call a Hamkins-Kidder machine **recursive** if it halts on all inputs and for all x and α, the real $c^T_\alpha(x)$ is (Turing-)recursive. We define $\mathbf{PSPACE}^\star$ to be the class of sets of reals that are decidable by recursive Hamkins-Kidder machines. Clearly,

$$\mathbf{PSPACE}^\star \subseteq \mathbf{PSPACE}^{\mathrm{HK},0}_{\omega^\omega} =: \mathbf{PSPACE}.$$

Question 10 *Can we prove that* $\mathbf{PSPACE}^\star \subseteq \mathbf{P} := \mathbf{P}^{\mathrm{HK}}_{\omega^\omega}$*?*

4 Nondeterministic computation

Savitch's Theorem [Pa94, Theorem 7.5] tells us that for Turing computations, nondeterminism does not increase space efficiency (in other words, $\mathbf{PSPACE} = \mathbf{NPSPACE}$).

In this section, we briefly look at the interaction between nondeterminism and our space complexity classes for Hamkins-Kidder machines and Koepke machines.

In [Sc03], Schindler defined the class $\mathbf{NP}_f$ without introducing a notion of nondeterministic Hamkins-Kidder computation. For reals x and y, we define as usual

$$y * x(n) := \begin{cases} y(k) \text{ if } n = 2k, \text{ and} \\ x(k) \text{ if } n = 2k+1. \end{cases}$$

We call a machine T a **$*$-time f machine** if it halts for all inputs and for all x and y, we have that $\mathbf{time}(y * x, T) < f(x)$. A set A is in $\mathbf{NP}_f$ if there is a $*$-time f machine T such that

$$x \in A \iff \exists y (T(y * x)\downarrow = 1).$$

If f is a constant function, we can replace the "$*$-time f machine" with a "time f machine".

Schindler's notion naturally connects to a notion of nondeterministic computation: a nondeterministic Hamkins-Kidder machine is a machine with the Hamkins-Kidder architecture but a instead of a program δ that is a function it has a relation that gives a set of allowed actions. A nondeterministic Hamkins-Kidder machine T is called a **nondeterministic time f machine** if all possible

T-computations with input x halt before time $f(x)$. A set A is nondeterministically decidable by a machine T if there is at least one possible T-computation that accepts x.

Proposition 11 *Let $f : \mathbb{R} \to \mathrm{Ord}$ be a function such that for all x, we have $\omega \cdot f(x) = f(x)$. Then the following are equivalent for a set of reals A:*

1. *$A \in \mathbf{NP}_f$, and*
2. *A is nondeterministically decidable by a nondeterministic time f machine.*

Proof. "$\Rightarrow$": Let T be a $*$-time f machine for deciding A. At input x, we use the first ω stages of the computation to generate arbitrary witnesses by the simple program described by "write either 0 or 1 and move on". Thus, a nondeterministic Hamkins-Kidder machine can produce at stage ω of the computation all possible values of y on the scratch tape. Now run T on the arrangement of x on the input tape and y on the scratch tape as if it were $y * x$ on the input tape. We know that T will reach the HALT state in less than $f(x)$ steps. So the entire computation uses less than $\omega + f(x) = f(x)$ steps and one of the branches of the computation accepts x.

"$\Leftarrow$": Let T be a nondeterministic time f machine deciding A. Then we can see the computation of T at input x as a finitely branching tree of height at most $f(x) < \omega_1$. The branching pattern in each branch b of the tree can be coded into a real y_b (the code is an element of WO coding the length of the computation in the branch b, thus identifying each step of the computation with a natural number, and a function assigning the behaviour of T at the computation step coded by n in the branch b).

We can now define a $*$-time f machine T^* as follows: on input $y * x$, the machine checks whether y is a code (in the sense of the previous paragraph) for a T-computation with input x, and –as long as it is–, follows this computation. Note that each step of the T-computation may take ω steps in the T^*-computation, as T^* has to search for the next command to execute in the code y. If at any point it turns out that y is not a code for a T-computation, the machine HALTs and returns 0.

If b is an accepting branch of T, then $y_b * x$ will be accepted by T^*, and for each y, the computation with input $y * x$ will take at most $\omega \cdot f(x)$ steps. q.e.d.

From the point of view of space constraints, it is easy to see that the proof of Proposition 11 is highly inefficient: the nondeterministic computation contains every single real as a potential snapshot of the scratch tape. This raises the third general question: can we come up with a nondeterministic algorithm that is space efficient?

More precisely, if T is a nondeterministic Hamkins-Kidder machine and b is a branch through its computation tree at input x with the sequence $\langle b_\gamma\,;\,\gamma < \xi\rangle$ of snapshots occurring on the scratch tape during the computation along b, we write

$$\ell_\gamma(b) := \min\{\eta\,;\, b_\gamma \in \mathbf{L}_\eta[x]\},$$
$$\mathbf{space}^0(b) := \sup\{\ell_\gamma(b)\,;\,\gamma < \xi\}, \text{ and}$$

$$\mathbf{space}^1(b) := \sup\{\omega_1^{b_\gamma}\,;\,\gamma < \xi\} + 1.$$

For $i \in \{0,1\}$, we say that a Hamkins-Kidder machine T is a **nondeterministic space^i f machine** if all possible T-computations with input x halt for all branches b, we have $\mathbf{space}^i(b) < f(x)$. We say that $A \in \mathbf{NPSPACE}_f^{\mathrm{HK},i}$ if it is decidable by a nondeterministic space^i f machine.

Question 12 *For what functions f do we have*

$$\mathbf{PSPACE}_f^{\mathrm{HK}} = \mathbf{NPSPACE}_f^{\mathrm{HK},i}?$$

Analogously, we can define nondeterministic space classes $\mathbf{NPSPACE}_f^{\mathrm{K}}$ for Koepke machines. Hamkins and Welch have noticed [HaWe03, Theorem 1.7] that in general, nondeterministic Hamkins-Kidder computation can be more powerful than deterministic Hamkins-Kidder computation. Their proof shows that the Hamkins-Kidder halting problem is in $\mathbf{NP}_{\omega_1}^{\mathrm{HK}}$. Combining this result with Propositions 6 and 11, we get that

$$\mathbf{PSPACE}_{\omega+1}^{\mathrm{K}} \subsetneq \mathbf{NPSPACE}_{\omega+1}^{\mathrm{K}}.$$

References

[CoLöTo05] S. Barry **Cooper**, Benedikt **Löwe**, Leen **Torenvliet** (*eds.*), CiE 2005: New Computational Paradigms, Papers presented at the conference in Amsterdam, June 8-12, 2005, Heidelberg 2005 [Lecture Notes in Computer Science 3526]

[DeHaSc05] Vinay **Deolalikar**, Joel D. **Hamkins**, Ralf-Dieter **Schindler**, $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{coNP}$ for Infinite Time Turing Machines, **Journal of Logic and Computation** 15 (2005), p. 577–592

[Ha02] Joel D. **Hamkins**, Infinite time Turing machines, **Minds and Machines** 12 (2002), p. 521–539

[Ha04] Joel D. **Hamkins**, Supertask Computation, *in:* [LöPiRä04, p. 141–158]

[Ha05] Joel D. **Hamkins**, Infinitary Computability with Infinite Time Turing Machines *in:* [CoLöTo05, p. 180–187]

[HaLe00] Joel D. **Hamkins**, Andy **Lewis**, Infinite time Turing machines, **Journal of Symbolic Logic** 65 (2000), p. 567–604

[HaLe02] Joel D. **Hamkins**, Andy **Lewis**, Post's problem for supertasks has both positive and negative solutions, **Archive for Mathematical Logic** 41 (2002), p. 507–523

[HaSe01] Joel D. **Hamkins**, Daniel E. **Seabold**, Infinite time Turing machines with only one tape, **Mathematical Logic Quarterly** 47 (2001), p. 271–287

[HaWe03] Joel D. **Hamkins**, Philip D. **Welch**, $\mathbf{P}^f \neq \mathbf{NP}^f$ for almost all f, **Mathematical Logic Quarterly** 49 (2003), p. 536–540

[Ko$_0$05a] Peter **Koepke**, Turing Computations on Ordinals, **Bulletin of Symbolic Logic** 11 (2005), p. 377–397

[Ko$_0$05b] Peter **Koepke**, Computing a model of set theory, *in:* [CoLöTo05, p. 223–232]

[$\mathrm{Ko}_0\mathrm{Ko}_1\infty$] Peter **Koepke**, Martin **Koerwien**, Ordinal computations, *to appear in:* Barry Cooper, Benedikt Löwe, Dag Normann (*eds.*), Mathematics of Computation at CiE 2005, special issue of the journal **Mathematical Structures in Computer Science**

[LöPiRä04] Benedikt **Löwe**, Boris **Piwinger**, Thoralf **Räsch** (*eds.*), Classical and New Paradigms of Computation and their Complexity Hierarchies, Papers of the conference "Foundations of the Formal Sciences III", Dordrecht 2004 [Trends in Logic 23]

[Pa94] Christos H. **Papadimitriou**, Computational Complexity, Reading MA 1994

[Sc03] Ralf **Schindler**, $\mathbf{P} \neq \mathbf{NP}$ for infinite time Turing machines, **Monatshefte der Mathematik** 139 (2003), p. 335–340

[We00a] Philip D. **Welch**, The length of infinite time Turing machine computations, **Bulletin of the London Mathematical Society** 32 (2000), p. 129–136

[We00b] Philip D. **Welch**, Eventually infinite time Turing machine degrees: Infinite time decidable reals, **Journal of Symbolic Logic** 65 (2000), p. 1193–1203

[We04] Philip D. **Welch**, Determinacy and Post's Problem for Infinite Time Turing Machines, *in:* [LöPiRä04, p. 223–237]

[We05] Philip D. **Welch**, Arithmetical Quasi-inductive definitions and the transfinite action of 1-tape Turing Machines, *in:* [CoLöTo05, p. 532–539]

[We06] Philip D. **Welch**, Non-deterministic halting times for Hamkins-Kidder Turing machines, THIS VOLUME