Math. Log. Quart. 43 (1997) 179 - 202

Mathematical Logic
Quarterly

Computation Models for Parameterized Complexity

Marco Cesati and Miriam Di Ianni

Department of Computer Science, University of Rome "La Sapienza",
via Salaria 113, 00198 Roma, Italy[1)]

Abstract. A parameterized computational problem is a set of pairs $\langle x, k\rangle$, where k is a distinguished item called "parameter". FPT is the class of fixed-parameter tractable problems: for any fixed value of k, they are solvable in time bounded by a polynomial of degree α, where α is a constant not dependent on the parameter. In order to deal with parameterized intractability, Downey and Fellows have introduced a hierarchy of classes $\mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots$ containing likely intractable parameterized problems, and they have shown that such classes have many natural, complete languages. In this paper we analyze several variations of the halting problem for nondeterministic Turing machines with parameterized time, and we show that its parameterized complexity strongly depends on some resources like the number of tapes, head and internal states, and on the size of the alphabet. Notice that classical polynomial-time complexity fails in distinguishing such features. As byproducts, we show that parameterized complexity is a useful tool for the study of the intrinsic power of some computational models, and we underline the different "computational powers" of some levels of the parameterized hierarchy.

Mathematics Subject Classification: 68Q05, 68Q25, 03D10, 68Q15.

Keywords: Turing machine, Computational complexity, Parameterized computational complexity.

1 Introduction

Parameterized complexity [2, 3, 4, 5, 6] is a new, powerful framework with which to address the different "parameterized behaviour" of many computational problems. Almost all natural problems have instances consisting of at least two logical items; many NP-complete problems [7] admit "efficient" algorithms for small values of one item (the *parameter*). For example, the NP-complete VERTEX COVER problem [7] admits a solving algorithm with running time bounded by $2n + 2^k$ [4]. Thus, for small values of the parameter k, an efficient, linear time solving algorithm exists. On the contrary, the best known algorithm for the similar NP-complete DOMINATING SET problem [7] has running time in $\Theta(n^{k+1})$. Although such an algorithm runs in polynomial time for any fixed k, its time requirement could be unacceptable even for not too large graphs G. Indeed, there seems to be a huge gap between the performance of one of the previous algorithms and the other: if the graph G has 100 vertices and

[1)] e-mail: {cesati, diianni}@dsi.uniroma1.it

the parameter k has value 3, then the VERTEX COVER's algorithm is roughly 500,000 times faster than the DOMINATING SET's one.

A parameterized problem is said to be *fixed parameter tractable* [4] if it admits a solving algorithm whose running time on instance (x, k) is bounded by $f(k) \cdot |x|^{\alpha}$, where f is an arbitrary function and α is a constant not depending on the parameter k. The class of fixed parameter tractable problems is denoted by FPT. Furthermore, a hierarchy of classes $\{\mathrm{W}[t]\}$, including likely fixed parameter intractable problems (modulo some unlikely collapse), has been defined [2]. While the "classical" complexity classes (P, NP, the polynomial hierarchy, and so on) are defined on the basis of specific models of computation (several variations of Turing machines), the definition of the W classes is more involved. For example, while NP is defined as the class of problems which are solvable by nondeterministic polynomial-time Turing machines, $\mathrm{W}[t]$ is the closure under "fixed parameter reductions" with respect to a "kernel problem". In other words, the classes of the W hierarchy do not have an immediate computational characterization.

A first contribution on this subject has been established in [1] by proving that the problem of deciding if a nondeterministic Turing Machine M, having exactly one tape and one head, halts in at most some parameterized number of steps is W[1]-complete. The previous result provides strong evidence to the conjecture $\mathrm{W}[1] \neq \mathrm{FPT}$. Indeed, a Turing machine is such a general model of computation that it seems not possible to guess the result of some nondeterministic computation without looking (in the worst case) at all computation's paths. Moreover, the result provides a characterization of W[1] in terms of a classical model of computation: all problems in W[1] have "candidate solutions" that are verifiable by a "short" computation of a simple Turing machine.

In this paper, we significantly extend that result; in particular, we consider Turing machines having many tapes and many heads, and we analyze the role played by several static resources (like the number of internal states and the size of the alphabet) in a bounded-length computation. In the classical setting almost all variations of the basic Turing machine model are polynomially-related [8], and therefore the classical complexity theory does not allow to separate such models. On the contrary, we show that parameterized complexity may be a very useful tool for such a kind of analysis. Indeed, we prove that different nondeterministic Turing machine models characterize different W classes. As a byproduct, we underline the different "computational powers" of some levels of the W hierarchy. The computational complexity of the problem of deciding if a nondeterministic Turing machine accepts a string x with a bounded length computation (the SNTMC problem) is somewhat related to the intrinsic "computational power" of the corresponding Turing machine models. Here, for "computational power" we should mean the "ability to do something *by using some resources*". For example, we prove that the general SNTMC problem is W[2]-hard, while SNTMC restricted to *total* Turing machines[2] is W[1]-complete. Of course, given a non-total device, it is straightforward to fill its transition table and to get an "equivalent", total device which is able to perform any task as the original one. But the non-total machine is more "powerful", since it is able to perform the same

[2] A Turing machine is said to be *total* if it has an applicable transition for any global configuration.

tasks *by using a smaller transition table*; indeed, the key idea of the parameterized reduction showing the W[2]-hardness of SNTMC consists in an "information hiding" trick that allows us to keep small the number of required transitions.

The paper is organized as follows. In Section 2 we address the formal setting of parameterized complexity by providing the basic definitions. In Section 3 we formally define the Turing machine model we use throughout the paper and the SNTMC problem we deal with. In Section 4 we prove the W[1]-completeness of several versions of the SNTMC problem: when restricted to total machines or to machines in which some of the number of heads, tapes, internal states and the size of the input are parameterized. In Section 5 we consider the general SNTMC problem and some other versions and we prove their membership to the W hierarchy and their hardness for W[2]. In Section 6 we consider two restrictions of the SNTMC problem (in which the number of heads and, respectively, the size of the alphabet or the number of internal states are parameters) and we prove their membership to FPT. Finally, in Section 7 we address some conclusions and we discuss some open problems.

2 The parameterized complexity setting

A *parameterized problem* is a set $L \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a fixed alphabet. A parameterized problem L is (uniformly) *fixed-parameter tractable* if there is a constant α and an algorithm Φ such that Φ decides if $\langle x, k\rangle \in L$ in time $f(k)|x|^\alpha$, where $f : \mathbb{N} \longrightarrow \mathbb{N}$ is an arbitrary function. The class of fixed-parameter tractable problems is called FPT.

Let L_1, L_2 be parameterized problems. We say that L_1 is (uniformly many : 1) *reducible* to L_2 if there is a constant α and an algorithm Φ which transforms $\langle x, k\rangle$ into $\langle x', g(k)\rangle$ in time $f(k)|x|^\alpha$, where $f, g : \mathbb{N} \longrightarrow \mathbb{N}$ are arbitrary functions, so that $\langle x, k\rangle \in L_1$ if and only if $\langle x', g(k)\rangle \in L_2$. The reduction is said to be *strong* if the function f is recursive.

The W classes are defined in terms of decision circuits. A logical circuit is of *mixed type* if it has gates of two kinds: *Small gates* (not-gates, and-gates and or-gates with bounded fan-in, usually fan-in 1 for not-gates and fan-in 2 for or- and and-gates), and *Large gates* (and-gates and or-gates with unbounded fan-in).

The *depth* of a circuit C is the maximum number of gates (small or large) on an input-output path in C. The *weft* of a circuit C is the maximum number of large gates on an input-output path in C. A family of decision circuits F has *bounded depth* (resp. *bounded weft*) if there is a constant K such that every circuit in the family F has depth (resp. weft) at most K.

Let F be a family of decision circuits (possibly having many different circuits with a given number of inputs). A *parameterized circuit problem* is associated to F:

$$L_F = \{\langle C, k\rangle \ : \ C \in F \text{ and } C \text{ accepts an input vector of weight } k\},$$

where the *weight* of a Boolean vector x is the number of 1's in the vector. A parameterized problem L belongs to W[t] if there exists a constant h such that L reduces to the parameterized circuit problem $L_{F(t,h)}$ for the family $F(t, h)$ of mixed type decision circuits of weft at most t and depth at most h. Finally, a parameterized problem L

belongs to W[P] if L reduces to the circuit problem L_F, where F is the set of all circuits (no restrictions). Of course, FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[P]. Many natural NP-hard or NP-complete problems have been found for the W classes [6, 2].

3 The basic Turing Machine model

A multi-tape, multi-head Turing machine is a physical device consisting of a finite control (a finite subset Q of a countably infinite set Q_∞) and $t+1$ (unidimensional) tapes (for some $t \geq 0$): an *input tape* (tape 0) and t *work tapes* (tapes $1, \ldots, t$). Each tape consists of an infinite sequence of *cells* indexed by integers. Each cell contains a symbol of a finite subset Σ of a countably infinite set Σ_∞. We assume that Σ always includes the *blank symbol* '$\square$'. On any tape i ($i = 0, \ldots, t$), there are h *heads* (heads $1, \ldots, h$); every head *scans* one cell at a time.

Initially, the input tape (tape 0) holds an input word x not containing blanks, while all the other tapes are filled by blanks. Each head on tape i ($i = 0, \ldots, t$) scans the same cell, and moreover all heads on tape 0 scan the cell containing the first symbol of the input word x (if this is not empty). We assume that, if $t > 0$, the input tape is a read-only one and that its heads cannot go beyond the two blanks immediately adjacent to x. Conversely, if $t = 0$, the unique tape of the Turing machine (tape 0) is also a work tape and, thus, it is not read-only. A machine with $t = 0$ and $h = 1$ is called *simple*.

The *transition rule* δ specifies the behaviour of the device: it is a subset of $Q \times \Sigma^{h(t+1)} \times Q \times \Sigma^{h \max\{t,1\}} \times \{-1, 0, +1\}^{h(t+1)}$ and each element represents an *instruction*. The first $1 + h(t+1)$ items of any instruction encode the current state and the symbols currently scanned under the heads, while the remaining ones encode the state to be reached by the machine, the symbols to be written by the heads, and the movements of the heads. The machine halts whenever there is no rule in δ whose first $1 + h(t+1)$ items match its internal state and the scanned symbols. If this happens in a particular state denoted as q_a, the input word is *accepted*. The Turing machine could also use a particular final rejecting state q_r. Of course, q_a and q_r are never in the first item of any instruction.

In a multi-head Turing machine it may happen that several heads scan the same cell: as in [9] we adopt a head-concurrency rule to be applied when such heads attempt to write different symbols: the head with the lower number wins out.

Definition 1. A *Turing machine* M is a 3-tuple $\langle t, h, \delta \rangle$, where $t+1$ represents the number of tapes, h denotes the number of heads per tape, and δ is the transition table.

Notice that in the above definition explicit references to Q and Σ are missing. In fact, the subsets of Q_∞ and Σ_∞ actually used by the machine (the "working sets") are implicitly defined by δ, while Q and Σ could include useless states and symbols. In the following, Q_M and Σ_M denote such working sets. When M is understood, we shall write simply Σ and Q. In order to simplify the proofs, it is convenient to make the assumption according to which the blank symbol '$\square$' cannot be written on any of the work tapes.

A *configuration* of M is a $(2t+3)$-tuple $c = \langle q, w_0, n_0, w_1, n_1, \ldots, w_t, n_t \rangle$, where q is the current state, each w_i ($i = 0, \ldots, t$) is a word over Σ_M, and n_i ($i = 0, \ldots, t$) is a

h-tuple of integers in $\{0, \ldots, 1+|w_i|\}$. The string w_i represents the non-blank portion of tape i. The convention that blanks cannot be written ensures that the non-blank portion of each tape is contiguous. Each value n_{ij} of the h-tuple n_i indicates that head j of tape i is scanning the n_{ij}th symbol in w_i. For a string x, let $x[i]$ denote the ith symbol in x if $1 \leq i \leq |x|$. If i is outside the indicated range, then by definition $x[i]$ denotes the blank symbol. The *initial configuration* on input x is defined to be $c_0(x) = \langle q_0, x, \langle 1, \ldots, 1\rangle, \varepsilon, \langle 0, \ldots, 0\rangle, \ldots, \varepsilon, \langle 0, \ldots, 0\rangle\rangle$, where q_0 is the start state and ε is the empty string.

Now we define a binary relation $\vdash_\delta$ (or $\vdash$, if δ is understood) on the configurations of δ: $c \vdash_\delta c'$ means the configuration c' is obtained from c by applying an instruction of δ; thus, c' is a *successor* of c and the sequence is called a *transition* or a *step*. We write $c \vdash_\delta^k c'$ if there is a k-steps sequence

$$c = c_0 \vdash_\delta c_1 \vdash_\delta \cdots \vdash_\delta c_k = c'.$$

The reflexive, transitive closure of the binary relation $\vdash_\delta$ is denoted $\vdash_\delta^*$ (or just $\vdash^*$).

3.1 The "short" computation problem

In this paper we study the parameterized complexity of a computational problem concerning acceptor nondeterministic Turing machines. In particular, we consider the problem related to the existence of "short" accepting computation paths, that is, accepting computation paths having bounded time. Formally, the SHORT NONDETERMINISTIC TURING MACHINE COMPUTATION (in short SNTMC) problem is defined as follows:

Instance: a (nondeterministic) Turing machine $M = \langle t, h, \delta\rangle$ and a word x over Σ_M.

Parameter: an integer k.

Question: is there an accepting computation path of M on input x requiring at most k steps?

CAI, CHEN, DOWNEY and FELLOWS [1] proved that SNTMC is W[1]-complete when restricted to simple machines. In this paper we significantly extend such results by studying several variations of this problem. We concentrate our attention on the role played by various resources and we define new parameters representing the limits imposed on the resources. In particular we are interested in bounding:

(T) the number of tapes t,

(H) the number of heads per tape h,

(Σ) the number of *non-blank* symbols $|\Sigma - \{\Box\}|$,

(Q) the number of *non-halting* internal states $|Q - \{q_a, q_r\}|$,

(I) the size of the input $|x|$.

We denote the different versions by "prefixing" the name of the basic problem by, respectively, T, H, Σ, Q, and I. For example, Σ-SNTMC represents the parameterized problem SNTMC having both the time and the size of the alphabet Σ as parameters. Moreover, we may consider the same problems having some resources fixed to particular values; as an example, Σ_2-SNTMC denotes the problem SNTMC such that

the parameter is the time k and the instances are restricted to Turing machines with binary alphabet.

The common idea of many proofs is that the resources are somewhat "interchangeable". For example, a Turing machine that uses many internal states can be easily simulated by a new Turing machine having just one internal state and an enlarged alphabet (a particular cell on some tape contains a symbol encoding the internal state of the simulated device): we economize on the number of internal states, yet the size of the alphabet increases.

However, such methods are often not applicable. For example, let us suppose that our goal is to simulate a single-tape, single-head Turing machine by using a single-tape, single-head Turing machine with just one internal state. Of course we can enlarge the alphabet, but this does not seem to be enough. Indeed, for example, in the next sections it is shown that SNTMC for simple machines with a parameterized number of internal states is fixed parameter tractable while we know that SNTMC restricted to simple machines (one tape and one head) is W[1]-complete. Thus, a simple machine with a bounded number of internal states has considerably less power than a simple machine with an arbitrarily large number of states.

4 Cases of W[1]-completeness

Recall the W[1]-completeness of SNTMC proved in [1] for simple machines. Our first theorem generalizes such a result to multi-tapes, multi-heads machines such that the number of heads and the number of tapes are parameters. Of course, since SNTMC restricted to simple machines trivially reduces to SNTMC for multi-tapes, multi-heads machines, the W[1]-hardness directly follows. Thus, the interesting part of our result concerns the membership to W[1]. Before proving it, we need the following lemma.

Lemma 1. H-I_0-T-SNTMC *belongs to* W[1].

Proof (sketch). In order to show membership in W[1], it suffices to show how an instance $\langle M, k, t, h\rangle$ of H-I_0-T-SNTMC can be translated into an instance $\langle C, k'\rangle$, where C is a circuit with weft 1 and constant depth. We arrange the circuit so that the input lines are partitioned into k' pools of variables such that exactly one line in each pool can be set to 1. The pools represent:

1. the ith transition, $1 \leq i \leq k$ (k pools);
2. for any head j and tape l, the head position at time i, $1 \leq i \leq k$, $1 \leq j \leq h$, $0 \leq l \leq t$ ($hk(t+1)$ pools);
3. the internal state at time i, $1 \leq i \leq k$ (k pools);
4. the symbol in cell j at time i, $1 \leq i, j \leq k$, $0 \leq l \leq t$ ($k^2(t+1)$ pools).

Thus, we take $k' = 2k + hk(t+1) + k^2(t+1)$. In order to force exactly one input to be set equal to 1 in each pool of input variables we add to the circuit, for each such pool of input variables and for each pair of variables x and y in the same pool, a small "not both" circuit representing $(\neg x \vee \neg y)$. Observe that an accepted weight k' input vector must have exactly one variable set to true in each of k' pools. Let n denote the number of input variables in this construction; then $n = k|\delta| + k|Q| + kh(t+1) + k(t+1)|\Sigma|$.

The remainder of the circuit encodes various checks on the legacy of the above choices. These consistency checks conjunctively determine whether the choices represent an accepting k-steps computation of M, much as in the proof of COOK's theorem, and they can be implemented so that each one involves only a bounded number b of input variables.

We have $O(n^b)$ "checking" circuits for consistency checks involving b values. All of the small "not both" and "checking" circuits feed into the single large output and-gate of C. The formal description of the details is laborious but straightforward. □

We are now ready to prove the full theorem.

Theorem 1. H-T-SNTMC *is* W[1]*-complete.*

Proof. We have already noticed that the problem is trivially W[1]-hard. In order to prove its membership to W[1], we reduce it to H-I_0-T-SNTMC. The assertion will follow from Lemma 1.

We first reduce H-T-SNTMC to H-I-T-SNTMC and then the latter problem to H-I_0-T-SNTMC.

Let $\langle M, x, k, \ldots \rangle$ be an instance of H-T-SNTMC, and let us consider the deterministic Turing machine T that, on input $\langle M, x, k, \ldots \rangle$, outputs the same machine M, a new input word x' which is the truncation of x to the first k symbols, a new parameter value equal to k (for the bound on the input length), and all the parameters of the original instance. Of course, the running-time of T is linear in the size of its input. We have to show that $\langle M, x, k, \ldots \rangle$ is a yes-instance of H-T-SNTMC if and only if $\langle M, x', k, k, \ldots \rangle$ is a yes-instance of H-I-T-SNTMC. Consider that if a computation path of $M(x)$ accepts in at most k steps, then this computation path reads at most k symbols of the input, and therefore there exists also a computation path of $M(x')$ which accepts in at most k steps. Conversely, if every computation path of $M(x)$ does not accept after k steps, then every computation path of $M(x')$ does not accept after k steps.

Finally, we show that H-I-T-SNTMC reduces to H-I_0-T-SNTMC. Consider an instance of H-I-T-SNTMC containing a nondeterministic Turing machine $M = \langle t, h, \delta \rangle$, an input word x and parameters k and $n = |x|$. Suppose $t > 0$, and consider the Turing machine $M' = \langle t+1, h, \delta' \rangle$ that (starting from empty input) writes $x[1], x[2], \ldots, x[n]$ on the additional work tape and then simulates M. It is easy to verify that a description of M' can be derived from a description of $\langle M, x \rangle$ in a linear number of steps, and that M, x, k, n are in a yes-instance of H-I-T-SNTMC if and only if M', k are in a yes-instance of H-I_0-T-SNTMC. The reduction in the case $t = 0$ is very similar. □

Is the behaviour of nondeterministic Turing machines having a "small" number of internal states "easily" predictable? The answer turns out to be negative. Indeed, we shall prove that nondeterministic machines with just one non-terminal internal state and two work tapes are able to solve W[1]-complete problems in parameterized polynomial time, and therefore that the corresponding "short nondeterministic computation problem" is W[1]-hard.

DOWNEY and FELLOWS [6] proved that CLIQUE (given a graph G and a parameter k, decide if G contains a complete subgraph of k nodes) is W[1]-complete. Actually, CLIQUE can be considered as the "prototype" of W[1]-complete problems.

For our goal, it is sufficient to show that, for any instance $\langle G, k\rangle$ of CLIQUE, it is easily derivable a two-tapes, single-head, single-state Turing machine which is able to decide $\langle G, k\rangle \in$ CLIQUE in time bounded by a function of k.

Theorem 2. CLIQUE *reduces to* H_1-I_0-Q_1-T_2-SNTMC.

Proof. Let $\langle G, k\rangle$ be an instance of CLIQUE, where $G = (V, E)$ is a graph and k is an integer. We show how to construct a nondeterministic single-head Turing machine $M = \langle 2, 1, \delta\rangle$ such that $Q_M = \{q_0, q_a, q_r\}$, and M can reach q_a in $k' = f(k)$ moves if and only if G contains a k-clique.

Let us assume that $V = \{1, \ldots, n\}$. Then Σ_M contains the symbols $\square$, \$, #, and five symbols for every node $i \in V$. Of course we can assume that $k \leq n$. Since the internal states q_a and q_r are final states, there is just one internal state which performs any operations (the initial state q_0). Moreover, M has two work tapes, but actually M uses just one tape square of the second tape.

The Turing machine M is designed so that any accepting computation path consists of three phases. In the first phase, M nondeterministically guesses k vertices of G and writes the corresponding symbols in the first k cells of tape 1. During the second phase, M checks if the symbols are pairwise distinct. Thus, Phase 2 requires the k symbols written on tape 1 to be scanned k times and it ends in $O(k^2)$ steps. Phase 3 verifies that the guessed vertices are a clique. As an example, we show the transitions in δ used during the third phase:

$\langle \square, \underline{v}, \$, q_0, \underline{v}, \underline{v'}, q_0, 0, +1, 0\rangle$ for $v \in \{1, \ldots, n\}$,
$\langle \square, \underline{v}, \underline{w'}, q_0, \underline{v}, \underline{w'}, q_0, 0, +1, 0\rangle$ for $v, w \in \{1, \ldots, n\}$, $v \neq w$, $\langle v, w\rangle \in E$,
$\langle \square, \underline{v}, \underline{w'}, q_0, \#, \#, q_r, 0, 0, 0\rangle$ for $v, w \in \{1, \ldots, n\}$, $v \neq w$, $\langle v, w\rangle \notin E$,
$\langle \square, \square, \underline{w'}, q_0, \#, \underline{w''}, q_0, 0, -1, 0\rangle$ for $w \in \{1, \ldots, n\}$,
$\langle \square, \#, \underline{w'}, q_0, \#, \underline{w''}, q_0, 0, -1, 0\rangle$ for $w \in \{1, \ldots, n\}$,
$\langle \square, \underline{v}, \underline{w''}, q_0, \underline{v}, \underline{w''}, q_0, 0, -1, 0\rangle$ for $v, w \in \{1, \ldots, n\}$, $v \neq w$,
$\langle \square, \underline{v}, \underline{v''}, q_0, \underline{v}, \$, q_0, 0, +1, 0\rangle$ for $v \in \{1, \ldots, n\}$,
$\langle \square, \#, \$, q_0, \#, \#, q_a, 0, 0, 0\rangle$.

It is easy to verify that Phase 3 terminates in $O(k^2)$ steps, that the total number of symbols is $5n+3$, and that the total number of transitions is at most $2n^2+(k+7)n+2$. Of course, M is easily derived from $\langle G, k\rangle$ in linear time in the size of G and k. It is trivial to verify that G has a k-clique if and only if $M(\varepsilon)$ accepts and, in every case, $M(\varepsilon)$ halts in at most $2k^2 + 3k + 4$ steps. □

Of course, such result can be easily generalized to any fixed number of work tapes greater than 2. By the converse, later in this paper we shall show that a single-head machine with less than two work tapes and one non-final internal state cannot solve a W[1]-complete problem (unless W[1]=FPT).

Corollary 1. H_1-I_0-Q_1-T_t-SNTMC *is* W[1]*-complete for* $t \geq 2$.

Proof. By the trivial reduction from H_1-I_0-Q_1-T_t-SNTMC to H-T-SNTMC, Theorem 1 and Theorem 2. □

We now prove that even the H_h-I_0-Q_1-T_0-SNTMC problem is W[1]-complete.

Theorem 3. H_h-I_0-Q_1-T_0-SNTMC *is* W[1]*-complete.*

Proof (sketch). Of course H_h-I_0-Q_1-T_0-SNTMC belongs to W[1], because it trivially reduces to H − T − SNTMC ∈ W[1] (Theorem 1). Now, let us show that the W[1]-complete H_1-I_0-T_0-SNTMC problem [1] reduces to H_2-I_0-Q_1-T_0-SNTMC.

Let $\langle M.k \rangle$ be an instance of H_1-I_0-T_0-SNTMC, where $M = \langle 0, 1, \delta \rangle$ is a nondeterministic Turing machine; let us consider the instance $\langle M', k' \rangle$ of H_2-I_0-Q_1-T_0-SNTMC such that $k' = 2k + 1$ and $M' = \langle 0, 2, \delta' \rangle$ is a nondeterministic Turing machine having $Q_{M'} = \{q_0, q_a, q_r\} \cap Q_M$ and, for $\$ \notin \Sigma_M$,

$$\Sigma_{M'} = \Sigma_M \cup \{\$\} \cup \{\underline{i} : 1 \leq i \leq k\} \cup \{\underline{q} : q \in Q_M\}.$$

The Turing machine M' operates in two phases: in the first one it executes $k+1$ right-moves with head 2, using the cell scanned under the other heads as a counter. In the second phase M' simulates the machine M by using the cell scanned under head 2 as a storage area which contains the current internal symbol of M. It is easy to verify that M' can perform these tasks with just one non-terminal state. It is straightforward to derive the details of δ'. □

4.1 SNTMC restricted to total machines is W[1]-complete

In order to extend the W[1]-completeness result to larger classes of Turing machines, we introduce the notion of "total" Turing machine. We say that a Turing machine is *total* if it cannot "hang up", that is, for every combination of scanned symbols and internal state, the transition table of the machine contains an applicable instruction. Formally:

Definition 2. A Turing machine $M = \langle t, h, \delta \rangle$ is *total* if its transition relation δ defines a total multi-valued function from the set $\Sigma_M^{h(t+1)} \times (Q_M - \{q_a, q_r\})$ to the set $\Sigma_M^{h \max\{t,1\}} \times Q_M \times \{+1, 0, -1\}^{h(t+1)}$.

A total Turing machine has a peculiar property: its transition table has a large number of instructions (in $\Theta(|\Sigma|^{h(t+1)} \cdot |Q|)$). Total machines seem to have somewhat less power than non-total ones: fix a non-total machine M and consider the total machine M' obtained from M by filling its transition table with all the missing instructions (any new instruction simply enters the reject state). Of course, M and M' accept the same language, yet the size of M is smaller, and thus M appears to be more powerful than M'.

We now show that SNTMC restricted to total Turing machines is still W[1]-complete. Notice that the number of tapes and the number of heads per tape are arbitrarily large and not bounded by any parameter; later in this paper we shall show that the same problem for non total machines is W[2]-hard, and thus not in W[1].

Lemma 2. SNTMC *restricted to total Turing machines reduces to* H_1-T_0-SNTMC.

Proof (sketch). We adopt essentially the classical "multi-tape and multi-head" to "single-tape and single-head" reduction as presented in [8], but we need some tricks in order to achieve the fixed parameter reduction.

Let $\langle M, x, k \rangle$ be an instance of SNTMC, where $M = \langle t, h, \delta \rangle$ is a nondeterministic Turing machine, δ is a total function and $x \in \Sigma_M^*$. Let us assume that $q = |Q_M|$, $s = |\Sigma_M|$ and $d = |\delta|$. We define a nondeterministic Turing machine $M' = \langle 0, 1, \delta' \rangle$

such that M' has $(h+1)(t+1)$ tracks on the tape grouped in $t+1$ sets of $h+1$ tracks (i.e., $(h+1)$ tracks for each of the M's tapes). One track in each set records the content of the corresponding tape of M and the other ones are blank, except for some markers in the cells that hold the symbols scanned by the heads of M. The finite control of M' stores the state of M, along with a count of the number of head markers to the right of M''s head. Moreover, the finite control of M' stores the ht symbols scanned by the M's heads.

Each move of M is simulated by a sweep from left to right and then from right to left by the head of M'. Initially, M''s head is at the leftmost cell containing a head marker. To simulate a move of M, M' sweeps right, visiting each of the cells with head markers and recording the symbol scanned by each head of M. When M' crosses a head marker, it must update the count of head markers to its right. When no more head markers are to the right, M' has seen the symbols scanned by each of M's heads, so M' has enough information for nondeterministically guessing the next nondeterministic move of M. Now M' makes a pass left, until it reaches the leftmost head marker. The count of markers to the right enables M' to tell when it has gone far enough. As M' passes each head marker on the leftward pass, it updates the tape symbol of M "scanned" by that head marker, and it moves the head marker one symbol left or right to simulate the move of M. Finally, M' changes the state of M (recorded in M''s control) to complete the simulation of one move of M. If the new state of M is accepting, then M' accepts.

We have to show that this construction is a fixed-parameter reduction. The alphabet of M' is $\Sigma_{M'} = (\Sigma_M \times \{0,1\}^h)^{t+1}$: a symbol in a cell encodes the symbols in the corresponding $t+1$ cells of M plus the corresponding h markers. Thus, the number of symbols in $\Sigma_{M'}$ is $s' = 2^{(t+1)h} s^{t+1}$. An internal state of M' encodes the internal state of M, the number of head markers at the right of the M''s head and the $h(t+1)$ symbols scanned by the M's heads. Thus, the number of internal states in $Q_{M'}$ is $q' = \mathrm{O}(qths^{h(t+1)})$. For every transition in δ, δ' has a corresponding transition. Moreover, we need the transitions performing the "sweeping" operations, and therefore the number of transitions in δ' is $d' = \mathrm{O}(d + q's') = \mathrm{O}(d + s^{(t+1)(h+1)} 2^{(t+1)h} qth)$ (observe that the "sweeping" operations are deterministic). Let $\|M\|$ be the length of a description of M. Therefore $\|M\| \geq d(\log q + th \log s) \geq q s^{(t+1)h} (\log q + th \log s)$, because δ is a total function. Then it is easy to verify that the length of a description of M' is $\|M'\| = \mathrm{O}(d'(\log q' + \log s')) = \mathrm{O}(th\|M\|^3)$.

It is easy to check that M' uses at most $6k^2$ steps to simulate k steps of M. Moreover, observe that $|x| = |x'|$, and therefore the above reduction is a fixed-parameter reduction from total-SNTMC to H_1-T_0-SNTMC. □

Lemma 3. H_h-T_t-SNTMC *reduces to* SNTMC *restricted to total Turing machines.*

Proof. Assume that $t \geq 0$ and $h \geq 1$ are fixed integers. Let $\langle M, x, k \rangle$ be an instance of H_h-T_t-SNTMC, where $M = \langle t, h, \delta \rangle$ and $x \in \Sigma_M^*$. Let us define the Turing machine $M' = \langle t, h, \delta' \rangle$, where $\Sigma_{M'} = \Sigma_M$ and $Q_{M'} = Q_M$. δ' is the "closure" of δ: it contains all the transitions of δ, and for every $\langle q, \sigma_1, \ldots, \sigma_{ht} \rangle \in Q_M \times \Sigma_M^{ht}$ which is not a "prefix" of a transition in δ, $\langle q, \sigma_1, \ldots, \sigma_{ht}, q_r, \sigma_1, \ldots, \sigma_{ht}, 0, \ldots, 0 \rangle \in \delta'$. Thus δ' is total.

Let us suppose that $\langle M, x, k\rangle$ is a yes-instance of H_h-T_t-SNTMC. Therefore there exists a deterministic computation path of $M(x)$ which accepts in at most k steps. By definition, the k transitions applied in such a deterministic computation path are in δ' too, and thus there exists a deterministic computation path of $M'(x)$ which accepts in at most k steps (just consider the same transitions). Now, suppose that $\langle M, x, k\rangle$ is a no-instance of H_h-T_t-SNTMC, and suppose that there exists a deterministic computation path of $M'(x)$ which accepts in at most k steps. Of course, the k transitions applied in such a deterministic computation path are in δ too, because every transition in $\delta' - \delta$ enters the final rejecting state q_r. This is a contradiction, because such a deterministic computation path of $M'(x)$ is also a legal deterministic computation path of $M(x)$, and therefore $\langle M, x, k\rangle$ should be a yes-instance.

Let s be the number of symbols in Σ_M (and $\Sigma_{M'}$), let q be the number of internal states in Q_M (and $Q_{M'}$), let d be the number of transitions in δ, and finally let d' be the number of transitions in δ'. Since we put in δ' at most one new transition for every possible "instruction's prefix", we have $d' \leq d + q\, s^{h(t+1)}$, and therefore (since by definition $q \leq d$ and $s \leq d$) $d' = O(d^{ht+h+1})$. Now recall that h and t are fixed constants, and observe that M' is derivable from M in $O(d^{ht+h+1})$ steps by simply looking at the transition table δ. □

Theorem 4. SNTMC *restricted to total Turing machines is* W[1]*-complete.*

Proof. By Lemma 2 and Lemma 3, SNTMC restricted to total Turing machines is equivalent to H_h-T_t-SNTMC. By Theorem 1, H_h-T_t-SNTMC is in W[1] (trivially H_h-T_t-SNTMC reduces to H-T-SNTMC), and, by the W[1]-completeness of SNTMC restricted to simple machines [1], H_h-T_t-SNTMC is W[1]-hard (trivially I_0-H_1-T_0-SNTMC reduces to H_h-T_t-SNTMC). □

5 Beyond W[1]

Let us consider here the most general version of the SNTMC problem: the number of tapes and the number of heads per tape are arbitrarily large integers, so as the size of the alphabet, the number of internal states, and the size of the input word. What about the parameterized complexity of such a problem? In order to show the membership of SNTMC to some class W[t], we should present, for any Turing machine M, input word x, and time bound k, a mixed type logical circuit of weft t and bounded depth able to accept a weight $f(k)$ Boolean input if and only if $M(x)$ accepts in time at most k. But there is a difficulty: while in the H-T-SNTMC case (Theorem 1) the input vector of weight $f(k)$ encodes both the k applied transitions and the k corresponding global configurations (so that the circuit only checks whether the input actually encodes a Turing machine computation), in this case we cannot adopt such an approach, because it does not seem possible to encode a global configuration of a Turing machine with arbitrarily large number of tapes and heads into an input word of weight $f(k)$. Now, the solution consists in codifying in the input *only the applied transitions* and not all the global configurations. This implies that the circuit must compute on its own the global configurations and the weft grows from 1 to k. Thus, the best result achieved so far is the following.

Theorem 5. SNTMC *is in* W[P].

Proof (sketch). Let $M = \langle t, h, \delta \rangle$ be a nondeterministic Turing machine, let $w \in \Sigma^*$ and let k be an integer. We show how to construct a circuit C with $k|\delta|$ input lines so that there exists a weight k input vector x such that $C(x) = 1$ if and only if there exists an accepting computation path of $M(w)$ having at most k steps.

The circuit C has a final large and-gate. The $k|\delta|$ input lines are partitioned into k blocks $T_1, \ldots, T_k$; any block T_i has $|\delta|$ input lines and encodes the transition applied in the ith step of the computation path.

The circuit C has the following gates:

For every block T_i $(1 \leq i \leq k)$ and for every state $q \in Q$, there are two large or-gates $g^Q_{\text{old}}(i, q)$ and $g^Q_{\text{new}}(i, q)$. The gate $g^Q_{\text{old}}(i, q)$ (respectively, $g^Q_{\text{new}}(i, q)$) has in input the input lines of the block T_i which correspond to the transitions having q as "old" (respectively, "new") state.

· Similarly, two large or-gates $g^\Sigma_{\text{old}}(i, l, j, \sigma)$ and $g^\Sigma_{\text{new}}(i, l, j, \sigma)$ are defined for every block T_i $(1 \leq i \leq k)$, for every tape $l \in \{0, \ldots, t\}$, for every head $j \in \{1, \ldots, h\}$ and for every symbol $\sigma \in \Sigma$ representing the transitions having σ as "old" (respectively, "new") symbol for tape l and head j, and one large or-gate $g^T(i, l, j, m)$, for every block T_i $(1 \leq i \leq k)$, for every tape $l \in \{0, \ldots, t\}$, for every head $j \in \{1, \ldots, h\}$ and for every move $m \in \{+1, 0, -1\}$, representing the transitions having m as move of the head j on tape l.

For every time $i \in \{0, \ldots, k\}$, for every tape $l \in \{0, \ldots, t\}$, for every head $j \in \{1, \ldots, h\}$ and for every tape position $p \in \{0, \ldots, k\}$, there is a small or-gate $pos(i, l, j, p)$. We shall enforce that $pos(i, l, j, p)$ has output 1 if and only if at time i the head j of tape l is scanning the position p.

For every time $i \in \{0, \ldots, k\}$, for every tape $l \in \{0, \ldots, t\}$, for every tape position $p \in \{0, \ldots, k\}$ and for every symbol $\sigma \in \Sigma$, there is a large or-gate $symb(i, l, p, \sigma)$. We shall enforce that $symb(i, l, p, \sigma)$ has output 1 if and only if at time i the cell of tape l in position p contains the symbol σ.

Time 0 corresponds to the initial configuration of $M(w)$; therefore actually $pos(0, l, j, p)$ and $symb(0, l, p, \sigma)$ are not gates, but hard-wired logical values (0's and 1's) which encode the initial global configuration.

The circuit C implements several consistency checks in order to ensure that the input lines encode a "legal" computation of $M(w)$. In particular:

1. At any time exactly one transition should be applied; thus for every block T_i $(1 \leq i \leq k)$, at most one input line must be set to 1 (hence an accepting weight k input vector *must* have exactly one input line set to 1 in any block T_i). Therefore, the final and-gate has in input the output of $(\neg l_1 \vee \neg l_2)$, for every pair of different input lines (l_1, l_2) in the same block T_i, for $1 \leq i \leq k$.

2. At any time $i \in \{1, \ldots, k\}$, the cell on tape $l \in \{0, \ldots, t\}$ scanned under head $j \in \{1, \ldots, h\}$ must be consistent with the transition applied at step i and the scanned cell at time $i - 1$. Therefore, $pos(i, l, j, p)$ is defined as follows:

$$(pos(i-1, l, j, p) \wedge g^T(i, l, j, 0)) \vee (pos(i-1, l, j, p-1) \wedge g^T(i, l, j, +1)) \vee (pos(i-1, l, j, p+1) \wedge g^T(i, l, j, -1)).$$

3. At any time $i \in \{1,\ldots,k\}$, the symbol contained on tape $l \in \{0,\ldots,t\}$ in the cell in position $p \in \{0,\ldots,k\}$ must be consistent with the transition applied at step i and the symbol contained in the same cell at time $i-1$. We assumed that a multi-head Turing machine implements a "priority model": if more than one head scan the same cell, then only the head with lowest index writes the new symbol. Therefore, $symb(i,l,p,\sigma)$ is defined as follows:

$$(symb(i-1,l,p,\sigma) \wedge \neg \bigvee_{j=1}^{h} pos(i-1,l,j,p))$$
$$\vee \bigvee_{r=1}^{h} [g^{\Sigma}_{\text{new}}(i,l,r,\sigma) \wedge pos(i-1,l,r,p) \wedge \neg \bigvee_{j=1}^{r-1} pos(i-1,l,j,p)].$$

4. At any time $i \in \{2,\ldots,k\}$, the "old" state of the applied transition must be equal to the "new" state of the transition applied at step $i-1$. Observe that it is sufficient to check that for any $q \in Q$, $g^{Q}_{\text{new}}(i-1,q) = 1$ implies $g^{Q}_{\text{old}}(i,q) = 1$, because we are granted that for every block T_i there is at most one $g^{Q}_{\text{old}}(i,q)$ gate and at most one $g^{Q}_{\text{new}}(i,q)$ gate having output 1 (for any block, at most one input line is set to 1). Therefore the final and-gate has in input the output of $(\neg g^{Q}_{\text{new}}(i-1,q) \vee g^{Q}_{\text{old}}(i,q))$, for every $i \in \{2,\ldots,k\}$ and $q \in Q$.

5. At any time $i \in \{0,\ldots,k-1\}$, the symbols contained in the scanned cells on any tape must be consistent with the "old" symbols of the transition applied at step $i+1$. Therefore the final and-gate has in input the output of

$$\neg pos(i-1,l,j,p) \vee \neg symb(i-1,l,p,\sigma) \vee g^{\Sigma}_{\text{old}}(i,l,j,\sigma),$$

for any symbol $\sigma \in \Sigma$, time $i \in \{i,\ldots,k\}$, tape $l \in \{0,\ldots,t\}$, head $j \in \{1,\ldots,h\}$ and position $p \in \{0,\ldots,k\}$.

6. The first transition of the computation must have q_0 as "old" state; therefore the output of $g^{Q}_{\text{old}}(1,q_0)$ is linked to the input of the final and-gate. Moreover, the computation must accept, and therefore the output of $g^{Q}_{\text{new}}(k,q_a)$ is linked to the input of the final and-gate (actually, the computation can accept in less than k steps; however it is easy to add several "no-operation" transitions having q_a as both "old" and "new" state).

It is straightforward to verify that the circuit C can accept a weight k input vector if and only if there is a k-steps accepting computation path of $M(w)$. Moreover, C has $O(k^2 ht|\Sigma||Q||\delta|^2) = O(k^2||M||^2)$ gates, and it is easy to construct a description of C in a squared number of steps in the size of M and w. □

Since there seems to be no easy improvement of Theorem 5, we now look for a W[t]-hardness result for SNTMC for $t > 1$. Next theorem shows that H_1-SNTMC is W[2]-hard: therefore, this problem is not in W[1], unless W[1] = W[2] (recall instead that H_1-T-SNTMC belongs to W[1], see Theorem 1).

The W[2]-hardness of H_1-SNTMC is established by showing a parameterized reduction from a known W[2]-complete problem, namely DOMINATING SET [5]: given a graph $G = (V,E)$ and a positive integer k as parameter, does there exist a subset V' of V such that $|V'| \leq k$ and for every vertex $u \in V$, the neighborhood $N[u]$ of u (i.e., u and the set of vertices adjacent to u) contains at least one element of V'?

Theorem 6. DOMINATING SET *reduces to* H_1-SNTMC.

Proof. Let us fix an instance $\langle G,k\rangle$ of DOMINATING SET, where $G = (V,E)$ is a graph and k is the parameter. In the following we assume that $V = \{1,\ldots,n\}$.

We consider the nondeterministic Turing machine $M = \langle n+1, 1, \delta \rangle$ with alphabet $\Sigma_M = \{\Box, \#, \$, \underline{1}, \ldots, \underline{n}\}$ and state set $Q_M = \{q_0, q_1, q_2, q_3, q_a\}$.

M is supposed to start with an empty input word, therefore the input tape 0 is useless. M operates in four phases:

Phase 1. M nondeterministically writes k vertices on tape 1, using one cell of tape 2 as a counter and internal state q_0. The formal description of the transitions of this phase is straightforward and, thus, omitted.

Phase 2. M fills the work tapes from 2 to $n+1$ with $k+1$ symbols '\$', using one cell of tape 1 as a counter. This phase is somewhat crucial, because M is not allowed to write the blank symbol '$\Box$'; therefore we must prepare a 'nice' portion of the work tapes.

$$\langle \Box, \underline{i}, \underbrace{\Box, \ldots, \Box}_{n}, q_1, \underline{i+1}, \underbrace{\$, \ldots, \$}_{n}, q_1, 0, 0, \underbrace{+1, \ldots, +1}_{n} \rangle \quad \text{for } i \in \{1, \ldots, k-1\},$$

$$\langle \Box, \underline{k}, \underbrace{\Box, \ldots, \Box}_{n}, q_1, \$, \underbrace{\$, \ldots, \$}_{n}, q_2, 0, +1, \underbrace{0, \ldots, 0}_{n} \rangle.$$

Phase 3. M scans the symbols on tape 1, and for every scanned vertex, write a symbol '#' on any tape $i+1$ such that $v \in N[i]$. Eventually (when M reads a blank on tape 1), M goes to next phase. Observe that every time M writes a symbol '#', then it moves the corresponding head to left; therefore every transition reads always '\$' in any tape i ($2 \leq i \leq n+1$). On the contrary, if M doesn't write a symbol '#' in some tape, then the corresponding head position does not change.

$$\langle \Box, \underline{v}, \underbrace{\$, \ldots, \$}_{n}, q_2, \underline{v}, \sigma_1^v, \ldots, \sigma_n^v, q_2, 0, +1, m_1^v, \ldots, m_n^v \rangle \quad \text{for } v \in V,$$

where, for every $i \in \{1, \ldots, n\}$,

$$\sigma_i^v = \begin{cases} \$ & \text{if } v \notin N[i], \\ \# & \text{if } v \in N[i], \end{cases} \quad \text{and} \quad m_i^v = \begin{cases} 0 & \text{if } v \notin N[i], \\ -1 & \text{if } v \in N[i], \end{cases}$$

$$\langle \Box, \#, \underbrace{\$, \ldots, \$}_{n}, q_2, \#, \underbrace{\$, \ldots, \$}_{n}, q_2, 0, +1, \underbrace{0, \ldots, 0}_{n} \rangle,$$

$$\langle \Box, \Box, \underbrace{\$, \ldots, \$}_{n}, q_2, \$, \underbrace{\$, \ldots, \$}_{n}, q_3, 0, 0, \underbrace{0, \ldots, 0}_{n} \rangle.$$

Phase 4. M attempts to move to the right the heads on tapes from 2 to $n+1$. Next, if M reads the symbol '#' on every such tape, then it accepts.

$$\langle \Box, \$, \underbrace{\$, \ldots, \$}_{n}, q_3, \#, \underbrace{\$, \ldots, \$}_{n}, q_3, 0, 0, \underbrace{+1, \ldots, +1}_{n} \rangle,$$

$$\langle \Box, \#, \underbrace{\#, \ldots, \#}_{n}, q_3, \#, \underbrace{\#, \ldots, \#}_{n}, q_a, 0, 0, \underbrace{0, \ldots, 0}_{n} \rangle.$$

Notice that δ contains $O(nk)$ transitions, and these are easily derived from a description of G in a squared number of steps in the size of V. Now we show that $M(\varepsilon)$ accepts in at most $3k+4$ steps if and only if $\langle G, k \rangle$ is a yes-instance. Suppose that $\langle G, k \rangle$ is a yes-instance; therefore there exists a subset V' of V with at most k

elements which is a dominating set. Then, consider the computation path of $M(\varepsilon)$ that writes on tape 1 all the elements of V' and nothing else (if V' contains less than k elements, the deterministic computation guesses some vertices of V' more than once). Let us consider now the tape $i+1$, where $1 \leq i \leq n$. It is easy to verify that at the end of Phase 3, for any tape $i+1$, the head is placed on the cell which is at the right of the last '#'. Then, in Phase 4, the computation path performs one right-move and the computation path accepts. It is easy to verify that the number of steps is at most $3k+4$.

Now, suppose that $\langle G, k\rangle$ is a no-instance; therefore for any subset V' of V with at most k elements, there is a vertex $u \in V$ such that $N[u] \cap V' = \emptyset$. We claim that $M(\varepsilon)$ does not accept. Indeed, every computation path of $M(\varepsilon)$ guesses a subset V' of V with at most k elements; therefore at the end of Phase 3, there is a tape $i+1$ which does not contain any #, since V' cannot be a dominating set. Thus, during Phase 4, the accepting final state cannot be reached, because M needs at least one # in every tape in order to successfully finish the phase. □

Notice that the transition function δ of the machine M derived in the proof of the previous theorem is not total, and indeed the main idea of the construction is avoiding to consider the exponentially-many configurations of scanned symbols different from all \$'s and all #'s. Actually, by Theorem 4, SNTMC restricted to total Turing machines is W[1]-complete. Moreover, the following result is easily derived.

Corollary 2. H_1-I_0-Q_1-Σ_2-SNTMC *is* W[2]-*hard.*

Proof. Our target is to reduce H_1-SNTMC to H_1-I_0-Q_1-Σ_2-SNTMC. Like in Theorem 1, H_1-SNTMC easily reduces to I_0-H_1-SNTMC.

Let us show that I_0-H_1-SNTMC reduces to I_0-H_1-Q_1-SNTMC. Let $\langle M, k\rangle$ be an instance of I_0-H_1-SNTMC, where $M = \langle t, 1, \delta\rangle$ is a single head nondeterministic Turing machine. Then we can easily construct a nondeterministic Turing machine $M' = \langle t+1, 1, \delta\rangle$ such that $Q_{M'} = \{q_0, q_a, q_r\} \cap Q_M$ and there exists an accepting computation path of $M(\varepsilon)$ with k steps if and only if there is an accepting computation path of $M'(\varepsilon)$ with k steps. We simply set $\Sigma_{M'} = \Sigma_M \cup \{\underline{q} : q \in Q_M\}$ and we arrange δ' so that M' uses one cell of the (additional) tape $t+1$ in order to encode the internal state of M. It is straightforward to derive the details of δ'.

It remains to show that I_0-H_1-Q_1-SNTMC reduces to I_0-H_1-Q_1-Σ_2-SNTMC. Let $\langle M, k\rangle$ be an instance of I_0-H_1-Q_1-SNTMC, where $k \in \mathbb{N}$ and $M = \langle t, 1, \delta\rangle$. We construct an equivalent Turing machine $M' = \langle t', 1, \delta'\rangle$, where $t' = t\lceil \log |\Sigma_M| \rceil$, $Q_{M'} = Q_M$, and $\Sigma_{M'} = \{\Box, 0, 1\}$. Let s be the cardinality of Σ_M and $\sigma_0, \sigma_1, \ldots, \sigma_{s-1}$ a fixed ordering of the symbols in Σ_M. Let us assume that $\sigma_0 = \Box$. The crucial idea for M' is to simulate each work tape of M (with alphabet size s) by using $\lceil \log s \rceil$ work tapes of M' (with alphabet size 2). If $t > 0$, then we do not care of tape 0, because our machines work on empty inputs. In the following, we assume also that $[i]_2$ denotes the sequence of symbols in $\{0,1\}$ which is the binary representation for i of length $\lceil \log s \rceil$. Therefore, the notations $[1]_2$ and $[0]_2$ should be regarded as equivalent to, respectively, $0, \ldots, 0, 1$ and $0, \ldots, 0$ of length $\lceil \log s \rceil$. For every transition $\langle \Box, \sigma_{i_1}, \ldots, \sigma_{i_t}, q, \sigma_{j_1}, \ldots, \sigma_{j_t}, q', 0, m_1, \ldots, m_t\rangle$ in δ, the transition table δ' contains $\langle \Box, [i_1]_2, \ldots, [i_t]_2, q, [j_1]_2, \ldots, [j_t]_2, q', 0, \underbrace{m_1, \ldots, m_1}_{\lceil \log s \rceil}, \ldots, \underbrace{m_t, \ldots, m_t}_{\lceil \log s \rceil}\rangle$. No other tran-

sition is in δ'. It is easy to verify that it is possible to obtain a description of M' in a linear number of steps on the length of a description of M (because $|\delta'| = |\delta|$). Moreover, it is easy to see that $\langle M, k\rangle$ is a yes-instance of I_0-H_1-Q_1-SNTMC if and only if $\langle M', k\rangle$ is a yes-instance of I_0-H_1-Q_1-Σ_2-SNTMC. □

Since H_1-SNTMC is W[2]-hard, it is natural to conjecture that even I_0-T_0-SNTMC is hard for W[2], because tapes and heads seem to be quite interchangeable. As an example, given an instance of I_0-H_h-T_0-SNTMC, it is easy to derive an equivalent instance of I_0-H_1-T_{h+2}-SNTMC. The new machine T simulates each step of the old one M in three phases: (1) it scans tape 1 (containing the encoding of M's tape together with the head positions) and records the scanned symbols on tapes $3, \ldots, h+2$ (each such tape corresponds to a particular head); then (2) T reads at once the symbols on tapes $3, \ldots, h+2$, it chooses a transition of M and then it write on the same tapes the new symbols and on tape 2 the new state and head's moves; finally, (3) T scans h times tape 1 and updates it according to the contents of the other tapes. It should be clear that T uses a small fixed number of internal states and that the size of its transition table is linear in the size of M. Conversely, given an instance of I_0-H_1-T_t-SNTMC, it is easy to derive an equivalent instance of I_0-H_{t+1}-T_0-SNTMC; the new machine T uses $t+1$ blocks of cells in its tape in order to record the contents of the $t+1$ tapes of the old machine M; each head of T scans a different block, and each block has exactly k cells.

However, the previous reasonings do not apply to this case, since they require that both the number of tapes and the number of heads are fixed or parameterized. Thus, in order to establish the hardness result, we are compelled to directly show that a single-tape, multi-head Turing machine is able to solve an instance of a W[2]-hard problem.

Theorem 7. I_0-T_0-SNTMC *is* W[2]-*hard.*

Proof. Let $\langle G, k\rangle$ be an instance of DOMINATING SET, where $G = (V, E)$ is a graph and k is the parameter. In the following we assume that $V = \{1, \ldots, n\}$.

We consider the nondeterministic Turing machine $M = \langle 0, n+2, \delta\rangle$ with one tape, $n+2$ heads, alphabet $\Sigma = \{\square, \$, \#, \underline{1}, \ldots, \underline{n}\}$ and set $Q = \{q_0, q_1, q_2, q_3, q_4, q_a\}$ of internal states. M operates in three phases and it is very similar to the machine described in Theorem 6. In this case, one head acts as a counter, one head is used to guess k vertices and to examine them in order to decide if they are a dominating set, and the remaining n heads simulate the behaviour of the head on each tape of the machine described in the previous theorem.

Phase 1. M nondeterministically writes k vertices on the tape, using the cell scanned under head 2 as a counter. In the meantime, M fills the tape at the left of the guessed vertices with \$'s.

$$\langle \underbrace{\square, \ldots, \square}_{n+2}, q_0, \underbrace{\$, \ldots, \$}_{n+2}, q_0, +1, 0, \underbrace{-1, \ldots, -1}_{n}\rangle,$$

$$\langle \square, \$, \underbrace{\square, \ldots, \square}_{n}, q_0, \$, \underline{1}, \underbrace{\$, \ldots, \$}_{n}, q_0, +1, 0, \underbrace{-1, \ldots, -1}_{n}\rangle,$$

$$\langle \square, \underline{i}, \underbrace{\square, \ldots, \square}_{n}, q_0, \underline{v}, \underline{i+1}, \underbrace{\$, \ldots, \$}_{n}, q_1, +1, 0, \underbrace{-1, \ldots, -1}_{n}\rangle \text{ for } v \in V,\ i \in \{1, \ldots, k\}.$$

Phase 2. M scans the vertices on the tape using head 1, and for every scanned vertex v, it moves the $(i+2)$th head to the right if and only if $v \in N[i]$ $(1 \leq i \leq n)$. Observe that any head $i+2$ (where $1 \leq i \leq n$) always scans a cell containing '\$'.

$$\langle \underline{v}, \$, \underbrace{\$, \ldots, \$}_{n}, q_1, \underline{v}, \$, \underbrace{\$, \ldots, \$}_{n}, q_1, +1, 0, m_1^v, \ldots, m_n^v \rangle \quad \text{for } v \in V,$$

$$\text{where for every } i \in \{1, \ldots, n\},\ m_i^v = \begin{cases} 0 & \text{if } v \notin N[i], \\ +1 & \text{if } v \in N[i], \end{cases}$$

$$\langle \$, \$, \underbrace{\$, \ldots, \$}_{n}, q_1, \$, \$, \underbrace{\$, \ldots, \$}_{n}, q_2, 0, 0, \underbrace{-1, \ldots, -1}_{n} \rangle.$$

Phase 3. M attempts to move to the left the heads from 3 to $n+2$; then it accepts if all heads $3, \ldots n+2$ scan \$'s.

$$\langle \$, \$, \underbrace{\$, \ldots, \$}_{n}, q_2, \$, \$, \underbrace{\$, \ldots, \$}_{n}, q_2, 0, 0, \underbrace{-1, \ldots, -1}_{n} \rangle \quad \text{for } i \in \{1, \ldots, r-1\},$$

$$\langle \$, \underline{r}, \underbrace{\$, \ldots, \$}_{n}, q_2, \$, \$, \underbrace{\$, \ldots, \$}_{n}, q_a, 0, 0, \underbrace{0, \ldots, 0}_{n} \rangle.$$

Observe that δ contains $O(nk)$ transitions, and these are easily derived from a description of G in a linear number of steps in the size of V. Like in Theorem 6, it can be now easily shown that $M(\varepsilon)$ accepts in at most $3k+4$ steps if and only if $\langle G, k \rangle$ is a yes-instance of DOMINATING SET. □

Since T_0-SNTMC reduces to I_0-Q_1-T_0-SNTMC (a similar reasoning to the one used in Theorem 1) the following result is easily derived.

Corollary 3. I_0-Q_1-T_0-SNTMC *is* W[2]*-hard.* □

6 Tractability results

Till now we have established the parameterized intractability of several variations of the basic "short nondeterministic computation" problem. We now consider some parameterizations that cause the problem to become fixed parameter tractable. Our first result concerns a machine M that (1) halts after a "small" number of steps, (2) has a "small" number of heads and tapes, and (3) writes only symbols of a "small" alphabet. In this case, M cannot solve too complex problems, since it can record on the tapes just a "small" amount of information during its computations. In order to prove such a result we first establish the following lemma.

Lemma 4. H-I_0-Σ-T-SNTMC *belongs to* FPT.

Proof. Let $\langle M, k, t, h, |\Sigma| \rangle$ be an instance of I_0-H-Σ-T-SNTMC. Then a bound on the total number of different global configurations involved in a k-step computation path of M on an empty input (in short, $M(\varepsilon)$) is $c = |Q| \cdot (|\Sigma|^k \cdot k^h)^{\max\{t,1\}}$. Let T be a deterministic Turing machine that, on input $\langle M, k \rangle$, simulates every computation path of $M(\varepsilon)$ for at most c steps. If T finds an accepting configuration, then T accepts, otherwise it rejects. It is easy to show that such a T decides every instance of H-I_0-Σ-T-SNTMC. However, we must describe how to construct T and discuss its running time.

To this aim, consider an oriented graph G (with at most c vertices) encoding the computation paths of $M(\varepsilon)$ such that every vertex represents a global configuration. T dynamically builds this graph; whenever T adds a node to the graph, it checks if the corresponding global configuration is an accepting one and if the corresponding global configuration already belongs to the graph. The algorithm simulating T is straightforward.

Every configuration of $M(\varepsilon)$ is a string of length $O((t+1)(h+k))$ over a suitable alphabet. $O(c^2)$ steps are enough for visiting G, and $O(ck)$ steps suffice to compare two configurations. Indeed, there can be only $O(c)$ cells on the tape between the encoding of the two configurations. Looking for the configurations reachable from a given one requires $O(|Q|^2|\Sigma|^{(t+1)h})$ steps (the maximum size of the transition table δ of M). Therefore the upper bound on the running time of the algorithm is $O(k|Q|^2c^5|\Sigma|^{2(t+1)h})$, i.e., $O(|\Sigma|^{(t+1)(5k+2h)}k^{5h(t+1)}|Q|^7)$. Hence H-$I_0$-$\Sigma$-T-SNTMC belongs to FPT. □

Theorem 8. H-Σ-T-SNTMC *belongs to* FPT.

Proof. By Lemma 4, in order to prove the theorem we just have to show that H-Σ-T-SNTMC reduces to H-I_0-Σ-T-SNTMC. The reduction is very similar to the one shown in Theorem 1 and thus omitted. □

We have already shown that H_1-Q_1-T_2-SNTMC is W[1]-complete. We still don't know if two work tapes are necessary for such a result or, vice versa, if we can reduce this number and still have the problem hard for W[1]. We start proving that H_1-Q-T_0-SNTMC reduces to H_1-Σ-T_0-SNTMC, and therefore that H_1-Q-T_0-SNTMC belongs to FPT. Then an easy corollary is that H_1-T_0-Q_1-SNTMC is also in FPT, and therefore we find a first, partial answer to the above question.

The main idea is the following. Suppose that a simple Turing machine has to decide if two cells do or do not contain the same symbol of the alphabet. The machine could perform this task in a very natural way: it reads the first symbol and enters an appropriate internal state; then it goes over the second symbol and decides. The crucial point here is that the machine needs one different internal state for every symbol of the alphabet. Thus, if a nondeterministic simple Turing machine M has a bound on the length of an accepting deterministic computation and on the number of internal states, then we can simulate it with a nondeterministic simple Turing machine having a bounded size alphabet.

In order to prove the fixed parameter tractability of H_1-T_0-Q-SNTMC, let us fix a nondeterministic simple Turing machine $M = \langle 0, 1, \delta \rangle$. Let $\Sigma = \Sigma_M$ be the alphabet of M and let $Q = Q_M$ be the set of internal states of M.

Now, we introduce a relation $\sim_1$ on Σ: given two symbols $\sigma_1, \sigma_2 \in \Sigma$, we say that $\sigma_1 \sim_1 \sigma_2$ if and only if, for every $\langle q, m \rangle \in Q \times \{+1, 0, -1\}$,

$$(\exists\, \sigma_1' \in \Sigma)\, \langle \sigma_1, q, \sigma_1', q_a, m \rangle \in \delta \;\Leftrightarrow\; (\exists\, \sigma_2' \in \Sigma)\, \langle \sigma_2, q, \sigma_2', q_a, m \rangle \in \delta.$$

It is easy to verify that $\sim_1$ is an equivalence relation. We denote by $[\sigma]_1$ the equivalence class with respect to $\sim_1$ which contains σ; moreover, $\Sigma/\!\sim_1$ is the set of all equivalence classes with respect to $\sim_1$.

Now we define by induction the equivalence relations $\sim_2, \ldots, \sim_k$ over Σ. Let l be an integer such that $1 < l \leq k$, and let $\sigma_1, \sigma_2 \in \Sigma$. We say that $\sigma_1 \sim_l \sigma_2$ if and only if for every $\langle q, q', m, [\sigma']_{l-1} \rangle \in Q^2 \times \{+1, 0, -1\} \times \Sigma/\!\sim_{l-1}$,

(i) if there exists $\sigma'_1 \in [\sigma']_{l-1}$ such that $\langle \sigma_1, q, \sigma'_1, q', m\rangle \in \delta$, then there exists $\sigma'_2 \in [\sigma']_{l-1}$ such that $\langle \sigma_2, q, \sigma'_2, q', m\rangle \in \delta$;

(ii) if there exists $\sigma'_2 \in [\sigma']_{l-1}$ such that $\langle \sigma_2, q, \sigma'_2, q', m\rangle \in \delta$, then there exists $\sigma'_1 \in [\sigma']_{l-1}$ such that $\langle \sigma_1, q, \sigma'_1, q', m\rangle \in \delta$.

It is easy to verify (by induction on l) that every $\sim_l$ is well-defined and is an equivalence relation on Σ. The following lemmata give explicit bounds on the number of elements of each set $\Sigma/\sim_l$.

Lemma 5. *$\Sigma/\sim_1$ contains at most $2^{3|Q|}$ equivalence classes.*

Proof. Consider a fixed $\langle q, m\rangle \in Q \times \{+1, 0, -1\}$. We first show that $\langle q, m\rangle$ can distinguish at most two equivalence classes, that is, $\langle q, m\rangle$ is a "bad" new for at most two symbols of Σ. Suppose that $\langle q, m\rangle$ separates different $\sigma_1, \sigma_2, \sigma_3 \in \Sigma$. Therefore we have for σ_1 and σ_2:

$$(1) \qquad (\exists \sigma'_1 \in \Sigma)\, \langle \sigma_1, q, \sigma'_1, q_a, m\rangle \in \delta$$

and

$$(2) \qquad (\forall \sigma'_2 \in \Sigma)\, \langle \sigma_2, q, \sigma'_2, q_a, m\rangle \notin \delta.$$

Because of equation (2), for σ_2 and σ_3 we have:

$$(3) \qquad (\exists \sigma'_3 \in \Sigma)\, \langle \sigma_3, q, \sigma'_3, q_a, m\rangle \in \delta.$$

This is a contradiction, because equations (2) and (3) imply that σ_1 and σ_3 cannot be separated by $\langle q, m\rangle$. Since the number of different $\langle q, m\rangle$ is $3|Q|$, there are at most $2^{3|Q|}$ $\sim_1$-equivalence classes on Σ. □

Let $\varphi : \mathbb{N}^2 \longrightarrow \mathbb{N}$ be the following recursively defined function:

$$\varphi(x, y) = \begin{cases} 2^{3y} & \text{if } x = 1, \\ \varphi(x-1, y) \cdot 2^{3y^2\varphi(x-1,y)} & \text{if } x > 1. \end{cases}$$

As an easy extension of the previous lemma, we can prove (by induction on l)

Lemma 6. *For any l, $1 \le l \le k$, $\Sigma/\sim_l$ contains at most $\varphi(l, |Q|)$ equivalence classes.*

Given a simple machine M and an integer k, it is easy to derive an FPT-algorithm computing $\Sigma/\sim_k$ as stated in the following lemma.

Lemma 7. *There exist two constants α, β and a deterministic Turing machine T such that, on input $\langle M = \langle 0, 1, \delta\rangle, k\rangle$, T computes $\Sigma/\sim_k$ in at most $O(k \cdot |Q|^\alpha \cdot |\Sigma|^\beta)$ steps.*

Proof (sketch). By induction on k. By Lemma 6, to compute $\Sigma/\sim_1$ it is sufficient to consider every $\langle \sigma_1, \sigma_2, q, m\rangle \in \Sigma \times \Sigma \times Q \times \{+1, 0, -1\}$ to check if there exist $\langle \sigma_1, q, \sigma'_1, q_a, m\rangle$, $\langle \sigma_2, q, \sigma'_2, q_a, m\rangle \in \delta$. Thus, T computes $\Sigma/\sim_1$ with $O(|Q|^4|\Sigma|^5)$ steps (recall that $\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{+1, 0, -1\}$). Similarly, due to the size of $\Sigma/\sim_i$, $1 \le i \le k$, and since $\Sigma/\sim_l$ can be computed from $\Sigma/\sim_{l-1}$ by considering all 4-tuples $\langle q, q', m, [\sigma']_{l-1}\rangle \in Q^2 \times \{+1, 0, -1\} \times \Sigma/\sim_{l-1}$, computing $\Sigma/\sim_k$ requires $O(k \cdot |Q|^5 \cdot |\Sigma|^8)$ steps. □

Let $M = \langle 0, 1, \delta\rangle$ be a nondeterministic Turing machine and let k be an integer. Let $x = x_0 x_1 \cdots x_m \in \Sigma^*$ and $1 \le l \le k$. Then denote by $x_{(l)}$ the word over $\Sigma/\sim_l$

such that $|x| = |x_{(l)}|$ and the ith symbol of $x_{(l)}$ is $[x_i]_l$. Moreover, let $\delta_{(l)}$ denote the subset of $Q \times \Sigma/\sim_l \times Q \times \Sigma/\sim_l \times \{+1, 0, -1\}$ such that

$$\langle [\sigma_1]_l, q, [\sigma_2]_l, q', m\rangle \in \delta_{(l)} \Leftrightarrow (\exists \sigma_1' \in [\sigma_1]_l)(\exists \sigma_2' \in [\sigma_2]_l)\, \langle \sigma_1', q, \sigma_2', q', m\rangle \in \delta.$$

Finally, let $M_{(l)}$ be the nondeterministic Turing machine $\langle 0, 1, [\delta]_l\rangle$; observe that the alphabet of $M_{(l)}$ is $\Sigma/\sim_l$.

Let c be a word over $\Sigma^* \times (\Sigma \times Q) \times \Sigma^*$ which encodes a global configuration of M. For example, $c = s_1 s_2 \cdots s_{i-1} \langle s_i, q\rangle s_{i+1} \cdots s_m$ encodes the global configuration such that the symbols on the non-blank portion of the tape are $s_1 \cdots s_m$, the symbol under the head is s_i and the internal state is q. For $1 \leq l \leq k$ denote by $c_{(l)}$ the word in $(\Sigma/\sim_l \cup (\Sigma/\sim_l \times Q))^*$ obtained by replacing every occurrence in c of a symbol s in Σ with the corresponding $[s]_l$ in $\Sigma/\sim_l$. It is easy to check that $c_{(l)}$ encodes a global configuration of $M_{(l)}$.

Finally, $M[c]$ denotes the nondeterministic computation of M starting from the global configuration c. We are now able to show that the Turing machines M and $M_{(k)}$ are "equivalent".

Theorem 9. *For every integer k, for every simple Turing machine M, and for every global configuration c, there is an accepting computation path of $M[c]$ with at most k steps if and only if there is an accepting computation path in $M_{(k)}[c_{(k)}]$ with at most k steps.*

Proof. Let M be a nondeterministic simple Turing machine, and let c be a global configuration of M. Let q be the internal state in the global configuration c, and let σ be the symbol under the head in c. We prove the theorem by induction on k and on the length of the actual computation path.

Let us assume that $k = 1$. Observe that if $M[c]$ accepts in 0 steps, then $q = q_a$, and therefore the theorem holds, since by definition $c_{(k)}$ is in the internal state q. Thus, we can assume that $q \neq q_a$. Let us suppose that there exists an accepting computation path of $M[c]$ with exactly one step. Therefore there exists a transition $\langle \sigma, q, \sigma', q_a, m\rangle \in \delta$. By definition of $\delta_{(1)}$, we have that $\langle [\sigma]_1, q, [\sigma']_1, q_a, m\rangle \in \delta_{(1)}$, and therefore there exists an accepting computation path of $M_{(1)}[c_{(1)}]$ with exactly one step.

Now let us suppose that there is an accepting computation path of $M_{(1)}[c_{(1)}]$ with exactly one step; then there exists a transition $\langle [\sigma]_1, q, [\sigma_2]_1, q_a, m\rangle \in \delta_{(1)}$. By definition of $\delta_{(1)}$, there exist $\sigma' \in [\sigma]_1$ and $\sigma_2' \in [\sigma_2]_1$ such that $\langle \sigma', q, \sigma_2', q_a, m\rangle \in \delta$. By definition of $\sigma \sim_1 \sigma'$, there exists $\langle \sigma, q, \sigma_3, q_a, m\rangle \in \delta$, and therefore there exists an accepting computation path of $M[c]$ with exactly one step.

Suppose now that the theorem holds for $k - 1$. We must show that there is an accepting computation path of $M[c]$ with $l \leq k$ steps if and only if there exists an accepting computation path of $M_{(k)}[c_{(k)}]$ with exactly l steps. We prove the assertion by induction on l.

The basis $l = 1$ can be proved in the same way as the case $k = 1$.

Now, assume the result holds for $l-1 \leq k-1$. Suppose first that there is an l-steps accepting computation path $c \vdash_\delta c_1 \vdash_\delta \cdots \vdash_\delta c_l$ of $M[c]$. Let $\langle \sigma, q, \sigma', q', m\rangle \in \delta$ be the first applied transition of the computation path. It is easy to see that $M[c_1]$ must

accept in $l-1 \leq k-1$ steps. Then, by inductive hypothesis on l, there exists an accepting computation path of $M_{(k)}[c_{1(k)}]$ with exactly $l-1$ steps. We have to show that there exists a transition in $\delta_{(k)}$ such that $c_{(k)} \vdash c_{1(k)}$. By definition of $\delta_{(k)}$, we know that $\langle [\sigma]_k, q, [\sigma']_k, q', m\rangle \in \delta_{(k)}$, and it is easy to verify that, if we apply this transition to $c_{(k)}$, then we obtain $c_{1(k)}$.

Finally, suppose that there exists an l-steps ($l \leq k$) accepting computation path $c_{(k)} \vdash c'_1 \vdash \cdots \vdash c'_l$ of $M_{(k)}[c_{(k)}]$. Let $\langle [\sigma]_k, q, [\sigma_2]_k, q', m\rangle \in \delta_{(k)}$ be the first transition of such a computation path. By definition of $\delta_{(k)}$, there exist $\sigma'_1 \in [\sigma]_k$ and $\sigma'_2 \in [\sigma_2]_k$ such that $\langle \sigma'_1, q, \sigma'_2, q', m\rangle \in \delta$. Since $\sigma'_1 \sim_k \sigma$, there exists $\sigma''_2 \in [\sigma'_2]_{k-1}$ such that $\langle \sigma, q, \sigma''_2, q', m\rangle \in \delta$, and let c_1 be the global configuration after the application of this transition. Now consider $M_{(k-1)}[c_{1(k-1)}]$. Observe that c'_1 and $c_{1(k-1)}$ have the same internal state and the same scanned position. Moreover, for almost every symbol $[s]_k$ in c'_1, the corresponding symbol in $c_{1(k-1)}$ is $[s]_{k-1}$. Indeed, every symbol $[s]_k$ is in c'_1 because the corresponding symbol in c is s, and therefore the corresponding in $[c_1]_{k-1}$ is $[s]_{k-1}$ (by definition). The unique exception is the symbol $[\sigma_2]_k$ in c'_1 which corresponds to the symbol $[\sigma''_2]_{k-1}$ in $c_{1(k-1)}$; this is not really an obstacle, because we know that $\sigma''_2 \sim_{k-1} \sigma'_2$ and $\sigma'_2 \sim_k \sigma_2$, and thus $[\sigma''_2]_{k-1} = [\sigma_2]_{k-1}$. Now, consider a transition $\langle [s_1]_k, q, [s_2]_k, q', m\rangle \in \delta_{(k)}$. By definition, there exist $s'_1 \in [s_1]_k$ and $s'_2 \in [s_2]_k$ such that $\langle s'_1, q, s'_2, q', m\rangle \in \delta$. But if $s' \in [s]_k$, then $s \sim_{k-1} s'$, and therefore $\langle [s_1]_{k-1}, q, [s_2]_{k-1}, q', m\rangle \in \delta_{(k-1)}$. Thus the $(l-1)$-steps accepting computation path of $M_{(k)}[c'_1]$ corresponds to a $(l-1)$-steps accepting computation path of $M_{(k-1)}[c_{1(k-1)}]$. By inductive hypothesis on l and k, there exists an accepting computation path of $M[c_1]$ with at most $l-1$ steps. Thus we have proved that there exists an accepting computation path of $M[c]$ with at most l steps. □

From the previous theorem we easily derive:

Corollary 4. *For any $k \in \mathbb{N}$, $\langle M, x, k\rangle$ is a yes-instance of* SNTMC *if and only if $\langle M_{(k)}, x_{(k)}, k\rangle$ is a yes-instance of* SNTMC.

Therefore we finally conclude:

Corollary 5. H_1-T_0-Q-SNTMC *belongs to* FPT.

Proof. Lemma 6, Lemma 7 and Corollary 4 show the existence of a parameterized reduction from H_1-T_0-Q-SNTMC to H_1-T_0-Σ-SNTMC, and therefore from Corollary 8, H_1-T_0-Q-SNTMC belongs to FPT. □

Till now we proved that H_1-Q_1-T_t-SNTMC is W[1]-complete for $t \geq 2$ and yet H_1-Q-T_0-SNTMC (and, thus, H_1-Q_1-T_0-SNTMC) is fixed parameter tractable. There is still an open case, namely the parameterized complexity of H_1-Q_1-T_1-SNTMC. The question essentially is: are the simple machines with a "small" number of internal states really equivalent (from a parameterized point of view) to the machines having "small" number of internal states, one input read-only tape and one work tape? The answer would be positive, whenever it were possible to remove from an instance of H_1-Q-T_1-SNTMC the input word. Indeed, in this case the input tape would be useless, and the equivalence would easily follow.

Theorem 10. H_1-T_1-Q-SNTMC *reduces to* H_1-I_0-T_1-Q-SNTMC.

Proof (sketch). Let $\langle M, x, k, p\rangle$ be an instance of H_1-T_1-Q-SNTMC, where x is a word over the alphabet $\Sigma_M - \{\square\}$ and $T = \langle 1, 1, \delta\rangle$ is a nondeterministic Turing

machine; its state set Q_M has at most p elements; tape 0 is the (read-only) input tape, and tape 1 is a work-tape. We show how to derive from $\langle M, x, k, p\rangle$ a nondeterministic Turing machine $M' = \langle 1, 1, \delta'\rangle$ having two tapes (one input tape and one work tape) such that $M'(\varepsilon)$ accepts in at most k steps if and only if $M(x)$ accepts in at most k steps. Notice that tape 0 of M' is useless. By the above assumptions, if $l = \min\{k, |x|\} + 1$, then the head on tape 0 in any k-step computation path of M scans always a cell which is in some position between 0 and l. Recall that, for any $j \in \{0, 1, \ldots, l\}$, $x[j]$ denotes either the jth symbol of x (if $1 \leq j \leq |x|$), or the blank symbol $\square$ (if $j = 0$ or $j > |x|$).

Let $Q_{M'} = ((Q_M - \{q_a, q_r\}) \times \{0, 1, \ldots, l\}) \cup \{q_a, q_r\}$ ($Q_{M'}$ has $(l+1)(p-2)+2$ elements) and let $\Sigma_{M'} = \Sigma_M$. The complete description of the transition table δ' of M' is the following. For every transition $\langle \sigma_1, \sigma_2, q', \sigma_2', q'', m_1, m_2\rangle \in \delta$ with $q'' \notin \{q_a, q_r\}$ we put in δ' the transition $\langle \square, \sigma_2, \langle q', j\rangle, \sigma_2', \langle q'', j + m_1\rangle, 0, m_2\rangle$ for any $j \in \{0, 1, \ldots, l\}$ such that $x[j] = \sigma_1$; otherwise, if $q'' \in \{q_a, q_r\}$, we put in δ' the transition $\langle \square, \sigma_2, \langle q', j\rangle, \sigma_2', q'', 0, m_2\rangle$. We must show that there is an accepting computation path of $M'(\varepsilon)$ with at most k steps if and only if there is an accepting computation path of $M(x)$ with at most k steps. Let c and c' be global configurations of, respectively, M and M'; we say that c and c' are *corresponding* if and only if

1. tape 0 of M contains the input x, and tape 0 of M' contains only blanks;
2. tape 1 of M and tape 1 of M' have the same contents;
3. the head position on tape 1 of M is equal to the head position on tape 1 of M';
4. the internal state of M is q', and the internal state of M' is $\langle q', j\rangle$, where j describe the head position on tape 1 of M.

Let us consider two corresponding configurations c and c', and let σ_1, σ_2 be the symbols scanned respectively on tape 0 and tape 1 of M in the global configuration c (therefore σ_2 is also the symbol scanned on tape 1 of M' in c'). We claim that there is an accepting computation path of $M[c]$ with exactly h steps if and only if there is an accepting computation path of $M'[c']$ with exactly h steps. The proof is by induction on the length of the computation paths.

Let us begin with $h = 1$. $M[c]$ accepts in one step if and only if there exists a transition $\langle \sigma_1, \sigma_2, q', \sigma_2', q_a, m_1, m_2\rangle \in \delta$, where σ_1 is either a blank (and then the head position on tape 0 is 0 or $|x|+1$), or a symbol of x (and then the head on tape 0 must be in position $j \in \{1, \ldots, l-1\}$ and the scanned cell on tape 0 must contain σ_1). However, this happens if and only if there exists a transition $\langle \square, \sigma_2, \langle q, j\rangle, \sigma_2', q_a, 0, m_2\rangle \in \delta'$, that is, if and only if there is an accepting computation path of $M'[c']$ with exactly one step.

Now, let us assume that the claim holds for $h - 1$. We know that there is an accepting computation path $c \vdash_\delta c_1 \vdash_\delta \cdots \vdash_\delta c_h$ of $M[c]$ with exactly h steps if and only if there exists $\langle \sigma_1, \sigma_2, q', \sigma_2', q'', m_1, m_2\rangle \in \delta$ which is applicable to c and produces c_1, and moreover there is an accepting computation path of $M[c_1]$ with exactly $h-1$ steps; that is (by construction of δ'), if and only if there exists a transition $\langle \square, \sigma_2, \langle q', j\rangle, \sigma_2', \langle q'', j + m_1\rangle, 0, m_2\rangle \in \delta'$ (where j is precisely the head position on tape 0 in c), and (by inductive hypothesis) there is an accepting computation path of $M[c_1']$ with exactly $h - 1$ steps (c_1' is the configuration corresponding to c_1). It is

easy to verify that the global configuration obtained by applying the transition of δ' to c' is exactly c'_1, and therefore the last sentence is equivalent to state that there is an accepting computation path of $M'[c']$ with exactly h steps.

It is easy to verify that it is possible to derive $\langle M', k, pk \rangle$ from $\langle M, x, k, p \rangle$ by using a linear number of steps in the size of the description of M and x. □

Now it is easy to get our goal:

Corollary 6. H_1-T_1-Q-SNTMC *belongs to* FPT.

Proof. By Theorem 10, H_1-T_1-Q-SNTMC reduces to H_1-I_0-T_1-Q-SNTMC. Trivially, H_1-I_0-T_1-Q-SNTMC reduces to H_1-I_0-T_0-Q-SNTMC (the input tape is useless!), and therefore H_1-T_1-Q-SNTMC reduces to the fixed parameter tractable problem H_1-T_0-Q-SNTMC (Corollary 5). □

7 Conclusions

In this paper we have proved some results about the parameterized complexity of the problem of deciding if a string x is accepted by a k steps computation of a given Turing machine M (k being the parameter).

In the classical complexity setting, the previous problem belongs to P if M is deterministic, and it is NP-complete if M is nondeterministic. This is the only computational distinction among Turing machines, in the sense that, independently from the number of heads, tapes and internal states, independently from the size of the alphabet, all deterministic Turing machines (and all non deterministic Turing machines) are polynomially related. Conversely, we have shown that, in the nondeterministic case, the parameterized complexity of the problem strongly depends on the above mentioned factors, and, thus, not all nondeterministic Turing machines are equivalent from a parameterized point of view.

A summary of the results proved in this paper is shown in Table 1.

problem	in	hard
SNTMC	W[P]	W[2]
H_1-I_0-Q_1-Σ_2	W[P]	W[2]
I_0-Q_1-T_0	W[P]	W[2]
(total)	W[1]	W[1]
H-T	W[1]	W[1]
I_0-H_1-T_0	W[1]	W[1]
I_0-H_1-Q_1-T_2	W[1]	W[1]
I_0-H_2-Q_1-T_0	W[1]	W[1]
H-Σ-T	FPT	—
H_1-Q-T_1	FPT	—

Table 1. The complexity of the SNTMC problem

There are a few remarks we want to discuss about our results. First, notice that, while Cai, Downey and Fellow's theorem about the W[1]-completeness of I_0-H_1-T_0-SNTMC holds for simple Turing machine, for the most general model of nondeterministic Turing machine we are able to prove only the membership to W[P]. Second,

it seems that simple machines are the more restricted model for which the W[1]-completeness result holds: indeed, if we introduce some further restriction, then the problem becomes included in FPT. On the other hand, if we try to generalize it by dropping any bound on the number of heads or tapes, then we obtain W[2]-hardness results.

It remains as an interesting open question the exact characterization of the problem for the models described in Section 5. Another interesting open question is whether there exists a natural resource for some Turing machine computation problem such that, if the resource is at most r, then the problem is complete for W[$f(r)$].

References

[1] Cai, L., J. Chen, R. G. Downey, and M. R. Fellows, On the parameterized complexity of short computation and factorization. To appear in the Archive for Mathematical Logic.

[2] Downey, R. G., and M. R. Fellows, Fixed-parameter intractability (extended abstract). In: Proceedings of the 7th IEEE Conference on Structure in Complexity Theory, Boston June 1992, pp. 36 – 49.

[3] Downey, R. G., and M. R. Fellows, Fixed-parameter tractability and completeness. Congressus Numerantium **87** (1992), 161 – 178.

[4] Downey, R. G., and M. R. Fellows, Parameterized computational feasibility. In: Feasible Mathematics II (P. Clote and J. Remel, eds.), Birkhäuser, Boston 1994.

[5] Downey, R. G., and M. R. Fellows, Fixed-parameter tractability and completeness I: Basic results. SIAM J. Computing **24** (1995), 873 – 921.

[6] Downey, R. G., and M. R. Fellows, Fixed-parameter tractability and completeness II: On completeness for $W[1]$. Theoret. Comp. Sci **141** (1995), 109 – 131.

[7] Garey, M. R., and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York 1979.

[8] Hopcroft, J. E., and J. D. Ullman, Introduction to Automata Theory, Languages and Computation. Addison-Wesley Publ. Comp., New York 1979.

[9] Lewis, H. R., and C. H. Papadimitriou, Elements of the Theory of Computation. Prentice-Hall, Englewood Cliffs, N.J., 1981.

(Received: March 30, 1996)