On Models of a Nondeterministic Computation

Mikhail N. Vyalyi[*]

Dorodnitsyn Computing Center of RAS,
Vavilova, 40, Moscow 119991, Russia
vyalyi@gmail.com

Abstract. In this paper we consider a nondeterministic computation performed by deterministic multi-head 2-way automata with a read-only access to an auxiliary memory. The memory contains additional data (a guess) and the computation is successful iff it is successful for some memory content.

Also we consider the case of restricted guesses in which a guess should satisfy some constraint.

We show that the standard complexity classes such as L, NL, P, NP, PSPACE can be characterized in terms of these models of nondeterministic computation. These characterizations differ from the well-known ones by absence of alternation.

Keywords: automaton, nondeterminism, language, complexity class.

The standard way to define a nondeterministic computation by an automaton or a Turing machine is to change a transition function by a transition relation. In a nondeterministic state of a computational device a computation branches into several computation paths.

There is another way to introduce a nondeterminism. Suppose that a computational device has an additional data (a *guess* or a *certificate* or a *proof of correctness*) and performs a deterministic computation operating with an input data and a guess data.

Sometimes these variants of introducing nondeterminism lead to equivalent computational models. The class NP, for example, can be defined in both ways using Turing machines.

If we restrict computational power then these variants may differ drastically. The aim of this paper is to investigate models of nondeterminism based on the second variant for multi-head 2-way automata.

It is well-known[1] that computation abilities of deterministic multi-head 2-way automata are equivalent to Turing machines with a logarithmically bounded auxiliary memory. In other words, they recognize languages from the class L.

Nondeterministic (in the sense of transition relation) multi-head 2-way automata recognize languages from the class NL. One can rewrite a definition of a

[*] The work is supported by the RFBR grants 08–01–00414, 09-01-00709 and the grant NS 5294.2008.1.

[1] O. H. Ibarra [12] attributed this result to A. Cobham and coauthors referring to an unpublished manuscript.

nondeterministic automaton in terms of guess data. For this purpose the 1-way read-only guess tape should be used.

Here we introduce a more general model of an auxiliary read-only memory (see definitions in Section 1). Guess data are stored in cells of a memory and at each moment of time an automaton has an access to the exactly one memory cell. Possible moves between memory cells form a directed graph (*a memory graph*). An automaton can choose between finite number of variants only. So the natural condition on the memory graph is a finite fan-out in each vertex (i.e. a memory cell).

The most natural variant of an auxiliary memory is a 2-way tape. This model appears to be very close to nonerasing nondeterministic stack automata (NENSA) [12,10]. The automata with the 2-way read-only guess tape recognize the same class of languages PSPACE as NENSA do.

A read-only memory may be useful in nondeterministic settings. Nevertheless, automata with a read-only nondeterministic memory can be related with a special variant of deterministic computation with an auxiliary memory, which is called WORM-automata (see Subsection 1.1). The WORM-automata with the 2-way guess tape are similar to the nonerasing deterministic stack automata (NEDSA) and also recognize the languages from the class PSPACE.

Also we introduce a nondeterministic computation with a restricted guess. An example of restricted guess is a *sparse guess*. Sparseness of a guess means that a guess tape contains the only one (or finitely many) non-empty symbol and the rest symbols stored on the tape are empty.

We focus our attention on a more restricted memory model, so-called 1.5-way tape. It was used in research of quantum automata [1]. For classic automata 1.5-way tape means an 1-way tape with a reset option, i.e. a possibility to make move into the initial cell from any memory cell.

The main results of this paper concern the 1.5-way tape memory.

The automata with the 1.5-way guess tape recognize the class PSPACE (Theorem 2 below) as the 2-way guess tape automata do. But the WORM-automata with this memory type recognize the class P only (Theorem 1). 1.5-way automata with sparse guesses recognize the class NP(Theorem 3). These results show that the 1.5-way guess tape is potentially more suitable to characterize various complexity classes.

An interesting feature of all these results is a formal absence of resource bounds in characterizations of resource-bounded classes such as P, NP and so on. It should be noted that there is a primary result of this sort: many heads are equivalent to logarithmic space. The rest of the results are based on this fact.

The main technical tool in study of the 1.5-way tape is calculations modulo polynomially bounded integer. These calculations can be performed on a logarithmic space. To compute a length of a part of the guess tape we use the simple algorithm: go along the part and increase a counter modulo p. The latter operation can be done on logarithmic space.

There are many results on characterizations of complexity classes in terms of some sort of automata. The classes L, NL, P, PSPACE have the well-known characterizations by deterministic, nondeterministic, alternating and synchronized alternating 2-way automata [5,11,7]. There are also characterizations of NP, the polynomial hierarchy and some other complexity classes in terms of alternating auxiliary stack automata [9].

Our results differ from these characterization because the models considered in this paper do not use alternation.

Some results in theory of tree-walking automata can be translated to our framework. In this case a memory model is a tree. For example, it follows from [3] that the automata with a read-only access to a tree memory recognize the languages from the class EXP.

It is worth to mention a paper [4], which contains the characterizations of P, NP and PSPACE in terms of nondeterminism only. The difference is in the nature of nondeterminism introduced. In [4] nondeterministic colorings of n-dimensional words are considered, where n is the input size. Contrary, our main results concern the case of 1-dimensional guess memory, which is potentially infinite. Using a potentially infinite tape makes more difficult an interpretation of the results in terms of descriptive complexity theory (see, e.g., the book [8]). For example, the results in [4] are directly related with Fagin's theorem that characterizes the class NP in terms of the second-order logic. To establish a similar relation to our characterization of the class NP one need specify suitably restricted infinite models. Up to the moment we know no way to implement this idea.

The rest of paper is organized as follows. In Section 1 we introduce our basic computational model: multi-head 2-way automata with a nondeterministic auxiliary memory. Section 2 contains results about the 1-way, the 1.5-way and the 2-way guess tapes. In Section 3 we introduce a model of a restricted guess and give characterizations of NP in terms of this model. In Section 4 we make some remarks on a more general memory model, which is called a monoid memory.

Details of proofs are omitted here due to space limitations. They can be found in the preprint [15].

1 Automata with an Auxiliary Read-Only Memory

In this section we provide definitions for a model of nondeterministic computation by automata with an auxiliary read-only memory. The definitions fix an informal idea explained in the introductory section.

Definition 1. A *memory model* is a directed graph (M, E), an initial cell $m_0 \in M$ and a marking map $g\colon E \to G$ from the edges of the graph to some finite set G. The marking map satisfies the following conditions:

– $g(u, v) \neq g(u, w)$ for $v \neq w$ (edges outgoing from the vertex can be distinguished by their marks);
– for each $u \in M$ and $a \in G$ there is an edge $(u, v) \in E$ such that $g(u, v) = a$.

In other words, the map g restricted to the set of edges outgoing from a vertex is a bijection.

For any finite alphabet Δ *a memory content* μ is a map $\mu\colon M \to \Delta$.

Definition 2. An *h-head automaton A with an auxiliary memory of model M* (an M-automaton for brevity) is characterized by

- a finite state set Q,
- a finite input alphabet $I = \Sigma \cup \{\triangleleft, \triangleright\}$, $\Sigma \cap \{\triangleleft, \triangleright\} = \varnothing$,
- a finite memory alphabet Δ,
- a transition function $\delta \colon Q \times I^h \times \Delta \to Q \times \{-1, 0, 1\}^h \times (G \cup \mathsf{0})$, which maps an $(h+2)$-tuple (the current state, symbols the heads on the input tape, the symbol in the current memory cell) to an $(h+2)$-tuple (a new state, a motion command for each head, a command of changing memory cell),
- an initial state $q_0 \in Q$,
- a set of accepting states $Q_a \subset Q$.

A configuration of the automaton A is an $(h+2)$-tuple $(q, i_1, \ldots, i_h, m)$ (the state, the positions of the heads, the memory cell). *A surface configuration* of the automaton A is an $(h+1)$-tuple $(q, i_1, \ldots, i_h)$.

The transition function defines a transformation on the set of the configurations. A *motion command* for a head is an element from the set $\{-1, 0, +1\}$ indicating the shift of the head along the input tape. A *command of changing memory cell* is an element of the marking set G or an empty command $\mathsf{0}$. In the case of a non-empty command $g \in G$ the automaton moves out the current memory cell along the edge marked by g. The empty command do not change the memory cell.

An automaton A operates on an input word $w \in \Sigma^*$ in natural way. We assume that the input word is extended by the endmarkers $\{\triangleleft, \triangleright\}$ indicating the beginning and the end of the word. The automaton starts from the initial state q_0, the initial position of each head is the leftmost symbol of the input word, the initial memory cell is m_0. On each step of operation the automaton changes the configuration as described above. The automaton stops iff either it reaches an accepting state or a head goes out the area bounded by the endmarkers.

Definition 3. An automaton A *accepts* an input word w iff for some memory content μ it stops in an accepting state.

The automaton *recognizes* the language L iff for any $w \in L$ it accepts w and for any $w \notin L$ it do not accept w.

We denote by M-NFA the class of languages recognized by automata with an auxiliary memory of model M.

1.1 Determinization

A nontrivial use of an auxiliary read-only memory is inevitably nondeterministic. Changing the read-only mode by the read-write mode in many cases leads to a broader language class. This contradicts an intuition that a deterministic model is weaker than a nondeterministic one.

In this subsection we describe a variant of deterministic use of an auxiliary memory of model M which gives a subclass of M-NFA. It is the WORM (write once, read many) mode. A WORM-M automaton should fill a new memory cell by a symbol when it enter the cell the first time. In further operation the automaton can not change the content of the cell.

Below we give a formal definition compatible with the above nondeterministic model.

Definition 4. A *WORM-memory automaton on memory model M* (a WORM-M automaton for brevity) is characterized by

– a finite state set Q,
– a finite input alphabet $I = \Sigma \cup \{\triangleleft, \triangleright\}$, $\Sigma \cap \{\triangleleft, \triangleright\} = \varnothing$,
– a finite memory alphabet $\Delta \cup \{\mathsf{void}\}$,
– a transition function δ, which maps a $(h+2)$-tuple (the current state, symbols of the input word under the heads, the symbol in the current memory cell) to a $(h+2)$-tuple (a new state, a motion command for each head, a command of changing memory cell),
– the initial state $q_0 \in Q$,
– the set of accepting states $Q_a \subset Q$,
– the set of writing states $Q_w \subset Q$.
– a filling memory function $\varphi \colon Q_f \to \Delta$,

At the start of operation all memory cells are void. A WORM-memory automaton operates in the same way as a nondeterministic M-automaton except the moments of entering a writing state. In that moment the filling function is applied to the current state of the automaton. If the current memory cell is visited at first time then the value of the filling function is assigned to the cell and the automaton continues operation by application of the transition function. An attempt to change the content of a cell visited before causes the error as well as an attempt to apply the transition function at a void cell. In the case of an error the automaton stops the operation and do not accept the input word.

So, during a successful operation the automaton enters a new memory cell in a writing state. Also note that if the automaton writes the non-void symbol d to the cell containing the symbol d then no error occurs. We call this property 'a freedom of writing the same'.

We denote by M-WORM the class of languages recognized by deterministic automata with an auxiliary WORM-memory of model M.

Lemma 1. *M-WORM $\subseteq$ M-NFA.*

The idea of the proof is simple: a nondeterministic M-automaton A simulating a WORM-M memory automaton B expects a memory content consistent with the operation of B. Details can be found in [15].

Due to Lemma 1 one can regard the WORM-M automata as a specific case of M-automata.

2 Complexity Classes Recognized by Automata with an Auxiliary Tape Memory

2.1 1-Way Tape

Let W_1 be an infinite 1-way tape (Fig. 1). The class W_1-NFA is just the class NL. Indeed, a W_1-automaton can read a symbol from the guess tape once. This symbol can be used to make a nondeterministic choice in a transition relation.

Note also that W_1-WORM = L because a symbol from the W_1-tape can not be reread.

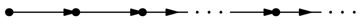

Fig. 1. 1-way tape W_1

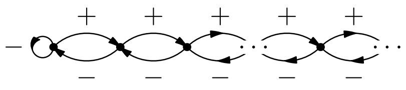

Fig. 2. 2-way tape W_2

2.2 2-Way Tape

Let W_2 be an infinite 2-way tape (Fig. 2). For graphs of fan-out > 1 we should also indicate the marking of edges. In the case of W_2 the marking is natural: mark '+' is placed on the edges going from a vertex n to the vertex $n + 1$, mark '−' is placed on the edges going to the opposite direction.

It was mentioned above that W_2-NFA = PSPACE because W_2-automata is almost the same as nonerasing nondeterministic stack automata and NENSA recognize the class PSPACE [12].

NENSA is able to make arbitrary nondeterministic transitions while an W_2-automaton should follow data read from the guess tape. It means that W_2-automata are weaker than NENSA, so W_2-NFA $\subseteq$ PSPACE. The reverse inclusion is valid even for WORM-W_2 automata. Indeed, a WORM-W_2 automaton is able to write a computational history of a Turing machine computation on a polynomially bounded space. For this purpose the automaton should move on distances polynomially bounded by the input size. This can be done by implementing polynomially bounded counters.

Thus, W_2-NFA $\subseteq$ PSPACE $\subseteq$ W_2-WORM $\subseteq$ W_2-NFA (the last inclusion is due to Lemma 1).

2.3 1.5-Way Tape

The memory model $W_{1.5}$ is pictured on the Fig. 3. Edges going to the right are marked by '+' and edges going to the initial vertex are marked by '−'.

Theorem 1. *$W_{1.5}$-WORM = P.*

The inclusion P $\subseteq$ $W_{1.5}$-WORM follows from the fact that a WORM-$W_{1.5}$ automaton A is able to simulate a WORM-W_2 automaton B on a polynomially

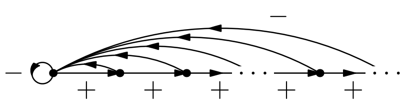

Fig. 3. 1.5-way tape $W_{1.5}$

bounded space by implementing a polynomially bounded counter that keeps a position of B on the W_2-tape.

The reverse inclusion follows from two simple observations. The first observations holds for general $W_{1.5}$-automata.

Lemma 2. *Let A be a $W_{1.5}$-automaton and $\#Q$ be the number of its states. Then any accepting computation of A includes no more than $\#Q$ moves to the initial cell.*

Proof. After each return move the automaton A scans the same tape content and its behavior is deterministic. So, if A starts the scan process from the same state twice it loops and never reach an accepting state.

Thus, the number of return moves is no more than the number of the states. □

The second observation is specific to the WORM-$W_{1.5}$ automata.

Proposition 1. *Let A be a WORM-$W_{1.5}$ automaton, h be the number of heads, n be the length of the input word w and $\#Q$ is the number of the states of A. If A accepts w then between two subsequent return moves the automaton visits no more than $n^h\#Q$ new cells.*

Proof. There are no more than $n^h\#Q$ surface configurations of A. If the automaton pass through more than $n^h\#Q$ new cells, some surface configuration occurs twice. It means that the automaton loops and moves to the right infinitely. □

These facts immediately imply that an operation of a WORM-$W_{1.5}$ automaton can be simulated by a Turing machine in polynomial time.

Thus the WORM-$W_{1.5}$-automata are weaker than the WORM-W_2 automata. As for nondeterministic automata, 1.5-way tape provides the same computational power as 2-way tape.

Theorem 2. $W_{1.5}$-NFA = PSPACE.

In one direction the inclusion is obvious:

$$W_{1.5}\text{-NFA} \subseteq W_2\text{-NFA} = \text{PSPACE}\,. \tag{1}$$

To prove the inclusion PSPACE $\subseteq W_{1.5}$-NFA we show that a $W_{1.5}$-automaton is able to check correctness of a computational history for a Turing machine computation on a polynomially bounded space. Configurations of the Turing machine are represented in special form using arithmetic encoding of binary words [13,14]. Namely, a word $w \in \{0,1\}^*$ is encoded by a positive integer $c(w)$ written in binary as $1w$.

A TM configuration ℓqar will be encoded by a 4-tuple $(c(\ell), q, a, c(r^R))$, where r^R denote the reversal of the word r.

It is easy to verify that the integers $c(\ell)$ and $c(r^R)$ before and after each step of the TM computation are related by equations

$$y = 2x,\ y = 2x + 1,\ x = 2y + 1,\ x = 2y\,, \tag{2}$$

where x is the old value and y is the new value of $c(\ell)$ or $c(r^R)$. The exact choice of a relation depends on the pair q, a and parities of $c(\ell)$, $c(r^R)$.

For a computation on a polynomial space it is sufficient to check relations (2) modulo polynomially bounded integers. A particular modular check can be done by a W_1-automaton that scans the description of the computational history from the left to the right. So a $W_{1.5}$-automaton can perform all modular checks jumping back to the initial cell before starting the next modular check.

The correctness of this procedure follows from the Chinese remainder theorem and the prime number theorem [2].

Details of the proof outlined above can be found in [15].

3 The Restricted Guess Case

In this section we introduce a generalization of the nondeterminism model. Namely, we will put a restriction on the form of a guess. The restriction can change a computational power of the model.

Definition 5. Let $\mathcal{T} \subseteq \Delta^M$ be a subset of memory contents. We say that a M-automaton A *accepts a word w with a $\mathcal{T}$-restricted guess* iff it accepts w operating on some memory content μ from the set $\mathcal{T}$.

We denote by $M(\mathcal{T})$-NFA the corresponding class of languages recognizable by M-automata with a $\mathcal{T}$-restricted guess.

Of course, in general $M(\mathcal{T})$-NFA $\not\subseteq$ M-NFA. For example, let $\mathcal{T}$ is the set of all valid computational histories of a Turing machines. Then $W_2(\mathcal{T})$-NFA contains all recursively enumerable languages.

We are interested in restrictions that describe subclasses of M-NFA. To guarantee the inclusion $M(\mathcal{T})$-NFA $\subseteq$ M-NFA it is sufficient to construct an automaton V that checks compatibility of memory content η in visited cells with the set $\mathcal{T}$. Compatibility means that η can be extended to some $\tau \in \mathcal{T}$. As an example of this kind of restriction we introduce *sparse guesses*.

Definition 6. Let $\Delta = \{0\} \cup \Delta'$. A *$k$-sparse guess* contains no more than k symbols from the Δ'.

We denote by U_k the set of k-sparse guesses.

Below we consider sparse guesses for tape memories.

3.1 Sparse Guesses for 1.5-Way Tape

The following facts can be verified easily by use of an informal idea of guess verification described above.

Lemma 3. $W_{1.5}(U_k)$-NFA $\subseteq W_{1.5}$-NFA *for any* k*.*

Lemma 4. $W_{1.5}(U_1)$-NFA $\subseteq W_{1.5}(U_k)$-NFA*.*

Proofs can be found in [15].

Now we give a characterization of the classes $W_{1.5}(U_k)$-NFA.

Theorem 3. $W_{1.5}(U_k)$-NFA $=$ NP *for* $k \geq 1$*.*

The proof of Theorem 3 is divided into two parts.

Lemma 5. NP $\subseteq W_{1.5}(U_1)$-NFA *for* $k \geq 1$*.*

An U_1-guess can mark a cell on the $W_{1.5}$-tape by a nonzero symbol. The distance d between the initial and the marked cell can be used to encode an information. The Chinese remainder theorem guarantee that one can encode polynomially many bits by residues modulo polynomially bounded primes. A $W_{1.5}$-automaton A recognizing an NP-language L expects that these bits form a computational history of a nondeterministic computation by a nondeterministic Turing machine recognizing the language L. To verify a guess the automaton A needs to extract a bit indexed by a prime p from the encoded data. This can be done computing the distance d modulo p.

Detailed exposition of the proof is contained in [15].

Lemma 6. $W_{1.5}(U_k)$-NFA $\subseteq$ NP *for any* k*.*

Due to Lemma 2 to find out the result of a $W_{1.5}$-automaton A operation on a U_k-guess one can divide the operation into polynomially many phases such that during each phase the $W_{1.5}$-automaton moves to the right and behaves like a W_1-automaton.

Note that the size of the set S of surface configurations of the $W_{1.5}$-automaton A is polynomially bounded by the input size. Changing a surface configuration after the reading of the symbol 0 is described by a map $\alpha_0 \colon S \to S$. Iterations of the map α_0 stabilize on some cycle: $\alpha_0^{i+\ell} = \alpha_0^i$ for sufficiently large i. an The length ℓ of the cycle is polynomially bounded. From these observations one can easily conclude that if the $W_{1.5}$-automaton A accepts on some U_k-guess then it accepts on a U_k-guess of exponential length, where the length of guess is the maximum of distances between the initial cell and cells marked by nonzero symbols.

So the positions of cells marked by nonzero symbols can be specified nondeterministically by NTM running in polynomial time. Given the positions one can compute the result of operation of the $W_{1.5}$-automaton A in deterministic polynomial time. For this purpose an algorithm of fast computation of matrix exponentiation can be applied.

Details of the proof can be found in [15].

Now Theorem 3 follows from Lemmata 4, 5, 6:

$$\mathrm{NP} \subseteq W_{1.5}(U_1)\text{-NFA} \subseteq W_{1.5}(U_k)\text{-NFA} \subseteq \mathrm{NP}\,. \tag{3}$$

3.2 Sparse Guesses for 2-Way Tape

U_k-guesses can be verified on the 2-way tape also.

Lemma 7. *$W_2(U_k)$-NFA $\subseteq W_2$-NFA for any k.*

This analog of Lemma 4 is proved in a straightforward way.

It is appeared that the class $W_2(U_1)$-NFA is rather weak.

Theorem 4. Aux2DC $\subseteq W_2(U_1)$-NFA $\subseteq$ Aux2NC $\subset$ P.

Here Aux2NC (Aux2DC) is the class of languages recognized by nondeterministic (deterministic) 2-way counter automata with a logarithmic auxiliary memory.

This effect is due to the absence of the root label in the initial cell. Using a non-zero symbol as the root label a W_2-automaton can simulate a deterministic counter automaton. Thus we obtain the first inclusion in Theorem 4.

To prove the second inclusion note that a position of the unique nonzero symbol on the tape can be found by a nondeterministic counter automaton (NCA) nondeterministically. After that the automaton can simulate the operation of a W_2-automaton A on a U_1-guess using the value of the counter to indicate the position of the W_2-automaton on the 2-way tape. This works well while the W_2-automaton is to the right of the cell marked by the nonzero symbol.

It is appeared that a behavior of the W_2-automaton while it moves between the initial and the marked cell can be simulated by the NCA nondeterministically using a logarithmic space. Details of the simulation can be found in [15].

The last inclusion in Theorem 4 follows from the Cook theorem [6]. The Cook theorem implies

$$\mathrm{Aux2PDA} = \mathrm{AuxN2PDA} = \mathrm{P}\,, \tag{4}$$

where Aux2PDA is the class of languages recognized by deterministic 2-way pushdown automata with a logarithmic auxiliary memory and AuxN2PDA is the class of languages recognized by nondeterministic 2-way pushdown automata with a logarithmic auxiliary memory.

For $k \geq 2$ the classes $W_2(U_k)$-NFA coincide with NP.

Theorem 5. *$W_2(U_k)$-NFA $=$ NP for $k \geq 2$.*

Note that Theorem 3 implies that $W_2(U_2)$-NFA $\supseteq$ NP because one nonzero symbol can be used to mark the initial cell and the other can be used for a simulation of a U_1-guess for a $W_{1.5}$-automaton.

The reverse inclusion can be proved in a way similar to the the proof of Lemma 6.

4 Monoid Memory

In this final section we briefly outline a natural extension of tape memories. The results mentioned below can be easily obtained by translation some well-known folklore facts to our settings.

Let G be a monoid generated by a set $G' = \{g_1, \dots, g_n\}$. Then the memory of type (G, G') is defined by the Cayley graph of the monoid M: the vertex set is G, an edge marked g_k goes from a vertex x to the vertex xg_k.

1-way and 2-way tapes are examples of monoid memory. It follows immediately from definitions that W_1-NFA $= (\mathbb{N}, \{+1\})$-NFA. Also it is easy to see that W_2-NFA $= (\mathbb{Z}, \{+1, -1)\})$-NFA.

There is a weak upper bound for the classes M-NFA of a monoid memory M.

Theorem 6. *Let M be a monoid. If the word problem for M is decidable then M-NFA $\subseteq \Sigma_1$, where Σ_1 is the class of recursively enumerable languages.*

The proof follows from the observation that a decision procedure for the word problem for the monoid M can be used to enumerate accepting computation histories of a M-automaton.

For many monoids and groups the bound of Theorem 6 is exact.

Take, for example, a $\mathbb{Z}^2$ memory. The generators of $\mathbb{Z}^2$ are chosen naturally: $(\pm 1, 0)$ and $(0, \pm 1)$.

The word problem for $\mathbb{Z}^2$ is decidable. So by Theorem 6 $\mathbb{Z}^2$-NFA $\subseteq \Sigma_1$.

On the other hand, a $\mathbb{Z}^2$-automaton is able to verify the correctness of computational history of an arbitrary Turing machine computation. The automaton expects a guess containing subsequent Turing machine configurations in subsequent rows of $\mathbb{Z}^2$. Correctness of computational history in this form is a conjunction of local conditions that can be verified by the automaton walking on $\mathbb{Z}^2$.

Thus $\mathbb{Z}^2$-NFA $= \Sigma_1$. As a corollary we get the following theorem.

Theorem 7. *Let G be a group with decidable word problem and $\mathbb{Z}^2$ is a subgroup of the group G. Then G-NFA $= \Sigma_1$.*

Acknowledgments

The author is deeply indebted to S. Tarasov for numerous valuable discussions and helpful references.

References

1. Amano, M., Iwama, K.: Undecidability on Quantum Finite Automata. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 368–375. ACM, New York (1999)
2. Bach, E., Shallit, J.: Algorithmic Number Theory. Efficient Algorithms, vol. I. MIT Press, Cambridge (1996)

3. Bojańczyk, M.: Tree-Walking Automata, `http://www.mimuw.edu.pl/~bojan/papers/twasurvey.pdf`
4. Borchert, B.: Formal Language Characterizations of P, NP, and PSPACE (2003) (submitted)
5. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. J. of ACM 28, 114–133 (1981)
6. Cook, S.A.: Characterization of Pushdown Machines in Terms of Time-Bounded Computers. J. of ACM 18, 4–18 (1971)
7. Geffert, V.: A Communication Hierarchy of Parallel Computations. Theoretical Computer Science 198, 99–130 (1998)
8. Grädel, E., Kolaitis, P., Libkin, L.: Finite Model Theory and its Applications. Springer, Heidelberg (2007)
9. Holzer, M., McKenzie, P.: Alternating and Empty Alternating Auxiliary Stack Automata. Theoretical Computer Science 299, 307–326 (2003)
10. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (1979)
11. Hromkovič, J., Karhumäki, J., Rovan, B., Slobodová, A.: On the Power of Syncronization in Parallel Computations. Discr. Appl. Math. 32, 155–182 (1991)
12. Ibarra, O.H.: Characterizations of Some Tape and Time Complexity Classes of Turing Machines of Multihead and Auxiliary Stack Automata. J. of Comp. and Sys. Sci. 5, 88–117 (1971)
13. Shen, A., Vereshchagin, N.: Mathematical Logic and Computation Theory. Languages and Calculi. In: MCCME, Moscow (1999) (in Russian)
14. Smullyan, R.M.: Theory of Formal Systems. Princeton Univ. Press, Princeton (1961)
15. Vyalyi, M.N.: On Models of a Nondeterministic Computation. Electronic preprint, arXiv:0811.2586 (2008)