PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article THEORETICAL COMPUTER SCIENCE

UDC 004.415.2:519.681
10.25209/2079-3316-2025-16-2-3-54

Complexity of Computations with Time Travel

Milad **Joudakizadeh**[1], Anatoly Petrovich **Beltiukov**[2]

[1, 2] Udmurt State University, Izhevsk, Russia.

[2]✉*belt.udsu@mail.ru*

Abstract. This paper explores a mathematical model of computation that can be interpreted as a computer capable of receiving data from future states of its own computational process. Under certain combinations of input data and programs, such computations may become infeasible due to emerging contradictions or lead to ambiguous outcomes. We investigate programs for which this process is always feasible and yields a unique result.

It is demonstrated that, in the absence of computational complexity constraints, such machines can output the value of any recursively decidable predicate within a fixed time after the computation begins—referred to as the response delay time—while the computation process must continue even after the result is produced. When the total runtime of these machines is bounded by a polynomial function of the input size, they precisely recognize languages belonging to the intersection of the NP and co-NP complexity classes, with the same constant response delay time in the aforementioned sense.

Possible practical implementations of such a computer are examined, including an analysis of operational protocols leveraging quantum annealing to select the appropriate computational process. It is shown that, with parallelization of the computational process, the class of problems solvable by these machines in polynomial time corresponds to the PSPACE complexity class.

Additionally, a mode of operation is studied in which these machines have direct access to input data. In this case, if the runtime is limited to a logarithmic function of the input size, the class of problems solvable by such a parallelized computer encompasses LOGSPACE

The findings of this study can be applied to develop new programming principles for nondeterministic computational machines, where data transmission from the future is employed instead of nondeterministic choices. (*Linked article texts in English and in Russian*).

Key words and phrases: Turing Machine, Computational Complexity, Recursive Sets, Nondeterministic Computation, Quantum Annealing, Time Machine, Artificial Intelligence

2020 *Mathematics Subject Classification*: 68Q09; 68Q10, 68Q15

For citation: Milad Joudakizadeh, Anatoly P. Beltiukov. *Complexity of Computations with Time Travel*. Program Systems: Theory and Applications, 2025, **16**:2(65), pp. 3–54. (*In English, in Russian*). https://psta.psiras.ru/read/psta2025_2_3-54.pdf

Introduction

This article presents an unconventional perspective on computational processes. What happens if a machine is allowed to peek into its own future and use this information to make decisions in the present? This idea, which at first glance seems inherently paradoxical, lies at the heart of the computational model under investigation—Turing machines with a backward-in-time communication channel. This model not only redefines the understanding of time in computation but also raises fundamental questions about causality and the interaction between past and future in computational processes.

Our machine is defined in such a way that it can receive information from future states of its computational process. This model represents a theoretical framework that introduces new ways of organizing computations. It allows obtaining the result of a complex computation before its completion and using this information. As with any time travel scenario, complications arise: transmitting data from the future may lead to contradictions and ambiguities.

In this article, we focus on cases where such problems can be avoided and demonstrate how this model can be implemented and applied to solve complex problems. This approach offers an alternative computational paradigm that connects time, causality, and computation, enabling new problem-solving strategies.

One of the results is that such a machine can compute the value of any recursively enumerable predicate in constant time. This means that tasks requiring significant time in classical models can be solved almost instantaneously (although the computational process itself must continue, and its total duration may still be substantial).

We also prove that if the total runtime of such machines is bounded by polynomials in the input size, they can recognize all languages in the intersection of NP and co-NP. These results indicate that the proposed model is not only theoretically interesting but also possesses significant computational power.

We also consider a modification of the machine that enables parallel data processing. It is shown that in this case, the class of problems solvable by such machines in polynomial time corresponds to PSPACE (i.e., polynomial time on the new machines is equivalent to PSPACE on ordinary Turing machines).

Another modification involves adding a mechanism for direct access to input data by addressing cells of the input tape using a special address tape. In this case, with runtime limited logarithmically in the input size, the class of problems solvable by our parallel computer includes LOGSPACE (here, LOGTIME also translates into corresponding space complexity). It is worth noting that this mode of operation does not entail excessive complexity and is feasible on conventional computers for reasonable input sizes.

From a practical standpoint, our work introduces new programming mechanisms. Instead of complex nondeterministic choices, which can be difficult to comprehend, we propose using data from the future, making the computational process more intuitively understandable. We also explore possible implementations of this model, including deterministic backtracking and quantum annealing.

In the first approach, systematic backtracking ensures that all possible paths are explored to prevent paradoxes. In the second approach, quantum adiabatic machines simultaneously explore multiple states of the computational process, selecting a self-consistent sequence of actions.

Thus, we propose a model that expands our understanding of computational processes and provides new approaches to programming.

1. Related Work

Over the past few decades, the idea of utilizing temporal paradoxes and time-traveling data transmission has attracted attention in theoretical computer science and physics. Many researchers have explored the possibility of extending classical computational models by incorporating mechanisms that allow information to be retrieved from the future, which is hypothesized to redefine traditional views on computational resources and complexity.

In the seminal work by Cook [1], the concept of NP-completeness was introduced, laying the foundation for further research in computational complexity and the reducibility of various problems to a universal problem. Expanding on these ideas, R. M. Karp [2] analyzed the reducibility of combinatorial problems, demonstrating that many seemingly disparate problems are interconnected.

The classic work by Garey and Johnson [3] became a fundamental reference for researchers studying computational complexity, presenting

a wide range of NP-complete problems and serving as a starting point for analyzing the limitations of traditional computational models.

In recent years, significant attention has been devoted to models that leverage physical effects to solve complex computational problems. Aaronson [4] investigated the potential of quantum computing for solving traditionally hard problems, showing that quantum methods can provide substantial advantages over classical algorithms.

Deutsch [5] introduced the concept of a universal quantum computer, marking a major milestone in quantum computing and influencing subsequent research on hybrid computational models.

Regarding models that violate classical causality, substantial contributions have been made in studies related to closed timelike curves (CTCs). Brun [6] demonstrated that the use of CTCs can significantly accelerate problem-solving. Moreover, Aaronson and Watrous [7] established the equivalence of quantum and classical computations in the presence of CTCs, highlighting the potential of such models to expand computational capabilities.

Beyond classical approaches, the idea of integrating time-travel-based data transmission with complexity theory has sparked interest. The works of Papadimitriou [8] and Arora and Barak [9] provide a deep analysis of the structure of computational problems, including the relationship between NP and co-NP classes. These studies serve as a foundation for analyzing how nonstandard computational models that utilize future data transmission can solve problems that are difficult for traditional models.

Classical works on quantum computing, such as the book by Nielsen and Chuang [10], as well as the algorithms of Grover [11] and Shor [12], illustrate how quantum methods can significantly accelerate the solution of certain problem classes.

More recent studies, such as those by Harrow and Montanaro [13] and Iqbal et al. [14], confirm the practical significance of quantum algorithms for solving complex computational problems.

An important aspect of this field is the concept of time-travel-based data transmission, which allows machines to predict their future states and use this information to adjust current computations. In this context, the work by Orlicki [15] explores a model of time-traveling machines and examines the problem of self-consistent halting. This approach extends

traditional models, offering new perspectives on the interaction of temporal flows in computation.

Furthermore, Gödel's contribution [16] demonstrated the theoretical possibility of closed timelike curves in cosmological solutions, providing additional physical justification for computational models involving future data transmission.

Studies by Bacon [17] and Wolpert [18] indicate that physical constraints on future prediction and feedback between temporal states significantly influence computational processes.

Finally, experimental research, such as the work by King et al. [19], shows that quantum systems with thousands of qubits can already simulate complex physical phenomena, raising hopes for the practical feasibility of computational models that exploit time-travel effects.

Thus, these works illustrate that the idea of future data transmission and temporal loops in computation has strong theoretical and practical foundations. Our time-traveling computation model naturally fits into this context, offering a fresh perspective on computational complexity and enabling the solution of problems that are intractable for traditional models.

2. Structure and Features of the Turing Machine with Time Data Transmission Channel

This section describes the construction of a Turing machine equipped with a time data transmission channel. This model extends the classical Turing machine by including a mechanism that allows data to be transmitted to previously executed steps of the computational process and to receive data transmitted from future steps. Let us consider the structure and features of such a machine.

2.1. Main Components of the Machine

The Turing machine with a time data transmission channel is a multi-tape machine (with one head for each tape) and includes the following elements:

Input Tape: This tape contains the input word that needs to be processed. At the beginning of the machine's operation, the input data is written to this tape.

Output Tape: The result of the machine's operation is recorded on this tape. For a recognition machine, the result can be represented as a binary value: 0 (no) or 1 (yes).

Multiple Working Tapes for performing necessary computations.

Communication Tape (Connection Tape): A special tape designed for transmitting data in time. The machine can write data to this tape for sending to the past and receive data from the future. To send and receive data, the machine must transition to special states for data transmission and reception, respectively.

Address Tape for Transmitted Data: This tape records labels that allow data to be retrieved by referencing these labels in requests. Special machine states are used for data transmission and retrieval.

In the transmission state, the contents of the communication tape along with the label are sent to the time data transmission channel. If data with such a label has already been sent earlier, it is considered that an error has occurred during the machine's operation.

In the reception state, the communication tape receives data from the transmission channel associated with the specified label. If such data was not found in the future, it is also considered that an error occurred during the machine's operation. For solving recognition tasks where errors may occur during operation, usage is prohibited.

Tape Alphabets: Each tape has its own alphabet of symbols that can be written on it.

Control Unit: At each step of operation, the control unit can be in one of the states of the state alphabet. The following special states are distinguished:

initial state (in which the machine begins the computational process),

final state (in which the machine's operation is completed),

data transmission state (to which transitioning sends data with the specified label to the data transmission channel),

data reception state (to which transitioning receives data from the data transmission channel with the specified label),

output state (to which transitioning sends the contents of the output tape to the outside world while the machine continues its operation), considering that the machine operates with a single write to the output tape, meaning that a repeated transition to the output state is an error in the machine's operation.

Program: A function (table) that maps the machine's state and the symbols visible to the heads on each tape to a new state, new symbols to be written on the tapes, and the directions of head shifts.

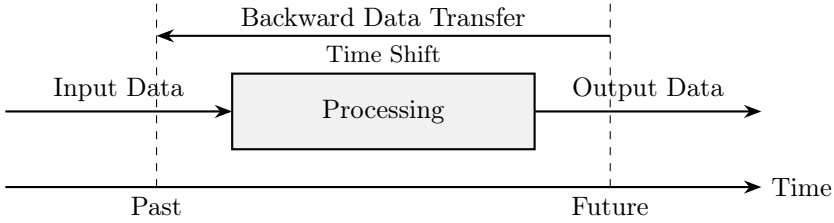

Figure 1. Schematic of reverse-time data transfer in a Turing machine with closed timelike curves

Computation Time: A sequence of steps. Each step performs one machine command or transmits/receives one symbol through the time data transmission channel (i.e., transmitting a long word to the data transmission channel takes as many steps as the length of that word). For a recognition machine, computation time must be finite.

In diagram Figure 1, the process of data processing with the possibility of transmitting information from the future to the past through a time shift is shown, implemented by closed time-like curves in Kerr metric.

2.2. Features of the Machine's Operation

The operation of the Turing machine with a time data transmission channel has several distinctive features:

Output of the result before the completion of computations: The machine can output the result to the output tape before the computation process is completed. This leads to a separation of two concepts of time:

response delay time,
total computation time.

Uniqueness and definiteness of computations: Due to the ability to obtain data from the future, the operation of the machine may become ambiguous or indeterminate. For example, ambiguity arises if the machine, having received some data from the future, writes the same data with the same label, which may lead to non-determinism in the process and, consequently, to different final results, depending on which data ends up being read and written. If the machine, having read data with one label, then necessarily writes different data with the same label, a contradiction arises, making the process

impossible, leading to indeterminacy. A machine is considered to recognize a certain property of the input data (i.e., computing the characteristic function of some set) if its operation is defined for all input data and the result is unique.

Programmability: At the end of the article, we will describe tools of algorithmic languages that will allow programming such actions not only for Turing machines but also for other computers.

2.3. Temporal Aspects of the Machine's Operation

A key feature of the machine is the ability to obtain data from the future, which leads to the following temporal effects:

Constant response delay time: Thanks to the mechanism of time data transmission, the machine can produce a simple result (YESor NO in constant time, regardless of computational complexity.

Total computation time: Despite the constant response delay time, the total computation time may be significantly longer, as the computation process must continue even after the result is output.

2.4. Explanation of Ambiguity and Indeterminacy

To illustrate the ambiguity and indeterminacy in the operation of the machine, consider the following example. Suppose the machine queries one bit from the future and chooses a branch of computations based on its value. If the bit value of 1 is received, the machine follows the first branch; if a value of 0 is received, it follows the second branch. Ambiguity in the computation process arises if, in each branch, the machine writes the same bit with the same label to the communication tape. This creates two possible paths of computation. It is important to distinguish between the ambiguity of computation and the ambiguity of the result. In an ambiguous computation process, the result may turn out to be the same.

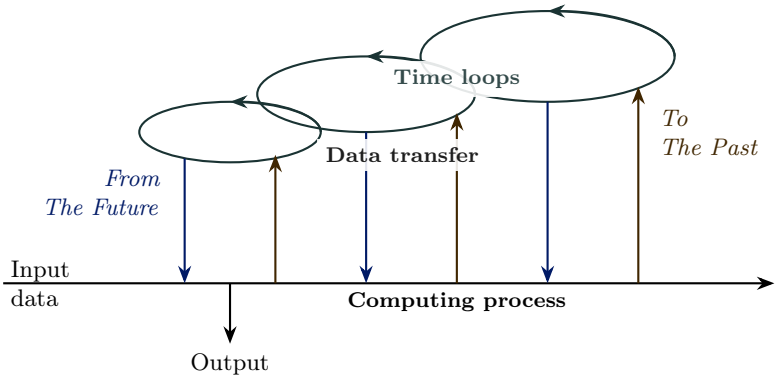

Figure 2. Schematic of temporal data flow in a Turing machine with closed timelike curves

If the machine reads one bit from the future and tries to write a contradictory bit to the communication tape with the same label, a contradiction arises, making such computation impossible.

Combinations of these situations may lead to some branches of the computation process, arising from internal ambiguity, subsequently becoming contradictory and thus being discarded. Only non-contradictory branches will remain. It may happen that all remaining non-contradictory branches lead to one final result. If this always occurs, we will consider such a machine correct for solving the recognition problem.

Thus, the described situations may lead to the following:

ambiguity, if several branches of computations end with different results,
indeterminacy, if no branch can complete without contradictions.

This demonstrates how the use of data from the future can affect the computational process.

3. Capabilities of Machines with Time Data Transmission

The scheme of data transmission in time for a Turing machine with closed time-like curves is presented in Figure 2.

It shows the main stages of information processing, the feedback mechanism through time loops, and the direction of data transmission.

In this section, we examine the computational capabilities of the Turing machine with a time data transmission channel. We present two theorems that demonstrate how the use of data from the future affects the computational complexity and classes of problems solvable by such a machine. These theorems show that the proposed model possesses properties that allow solving problems inaccessible to classical computational systems.

3.1. Recognition of Decidable Sets

Theorem 1. *A Turing machine with a time data transmission channel recognizes all recursively decidable sets with constant response delay time.*

Proof. The proof is based on the machine's ability to use data from the future to predict the results of computations. Let us recognize the membership of input strings in a given recursively decidable set. There exists a standard recognizing Turing machine for this set. From this machine, we will construct the required Turing machine with a time data transmission channel that will operate as follows.

At the start of its operation, the machine requests the contents of the communication tape labeled 0. This value is output as the result of its operation. Then, the process of the standard Turing machine recognizing membership in the given set is initiated. When the recognition is complete, its result is transmitted to the past with the label 0. This ensures that in the past, the machine provided the correct result.

Now, let us assume we have the original machine with a time data transmission channel that recognizes the membership of input strings in some set. By definition, the total running time of such a machine is always finite. We will construct a new standard Turing machine that recognizes membership in this set, which enumerates all possible strings that could serve as a partial protocol of the original machine's operation (chains of instantaneous descriptions), in order of their lengthening.

The machine checks for each string whether it is a correct completed protocol of the original machine's operation. As soon as such a protocol is found, the result of the original machine's operation is output as the result of the new machine. Since the operation of the original machine, by definition, is definite and unique, the new machine always provides the correct answer regarding the membership of the input string in the given set. □

3.2. Recognition of Sets in NP∩ co-NP

Theorem 2. *A Turing machine with constant response delay time and polynomial total computation time recognizes exactly all sets in* NP∩ co-NP

Idea of the Proof. By interpreting the reception of data from the future as branching into several paths, we construct a machine that operates in polynomial time and ensures the consistency of results across all branches of computations.

4. Proof of Theorem 2

In this section, we will present the proof of Theorem 2, formulated in the previous section. First, we will reformulate the operation of our machine in terms of nondeterministic computations. This will allow us to transform the operation of our machine into the operation of standard nondeterministic Turing machines and vice versa. Since the mechanism for transitioning to constant response delay has been described in the proof of the previous theorem, we will initially prove Theorem 2 without mentioning this, which we will do.

4.1. Transition to Nondeterministic Computations

The idea is to interpret the reading of each data symbol on the communication tape, received from the future by some label, as branching in the computation process. Each branch corresponds to one of the possible values of that symbol. When the machine reaches the moment in time denoted by this label, it checks whether the assumed read symbol matches the transmitted one. If the value of the symbol recorded in the communication tape cell matches the expected value in the current branch, the computations continue. Otherwise, the computation branch ends due to a contradiction.

A correct recognizing machine must provide the same answers (either 0 — NO or 1 — YES on all consistently terminating branches for a given input word. Next, we will transform our original machine with time data transmission into a nondeterministic one, as shown above, and then split it into two nondeterministic Turing machines, handling the aforementioned contradictions as follows:

(1) The first machine accepts the recognized set, returning 1 when the original machine outputs 1, and returns 0 in all other cases (including contradictions).

(2) The second machine accepts the complement of this set, returning 1 when the original machine outputs 0, and returns 0 in all other cases (including contradictions).

Thus, if the original machine with time data transmission outputs YES then the first of the resulting machines outputs YESon some branches of the computation, while the second always outputs NO Conversely, if the original machine outputs NO then the first always outputs NO while the second outputs YESon some branches of the computation. Thus, the first of the constructed deterministic machines accepts the recognizable set itself, while the second accepts its complement. It is easy to see that the polynomial time bound is preserved in the specified transformation, as all performed restructurings give at most polynomial slowdown. So, if the original machine recognized the given set in polynomial time, the newly constructed machines operate in the same time. This means that the originally recognized set belongs to NP∩ co-NP

4.2. Constructing a Machine with Time Data Transmission from Nondeterministic Machines

The reverse statement is also true: if there exist two nondeterministic machines, bounded by polynomial time, one of which accepts some set and the other accepts its complement, we can construct a single machine with time data transmission from the future that recognizes this set and operates in polynomial time. This machine operates as follows:

(1) At the beginning of the process, it guesses which of the two nondeterministic machines to run.

(2) It then executes the chosen machine.

(3) If the first selected machine outputs 1, the computation branch ends with the result 1.

(4) If the first selected machine outputs 0, the computation branch leads to a contradiction.

(5) If the second selected machine outputs 1, the computation branch ends with the result 0.

(6) If the second selected machine outputs 0, the computation branch leads to a contradiction.

By assumption, exactly one of these machines must yield a positive result. Thus, our machine always provides a definite answer: either 1 or 0.

4.3. Constant Response Delay Time

Constant response delay time is achieved because the decision about the result of the computations is made at the very beginning of the process. The machine requests data from the future and uses it to immediately output the answer. Meanwhile, the computation process must continue. □

5. Implementation Questions

This section discusses possible approaches to the practical implementation of a Turing machine with a time data transmission channel. Although such a machine is a theoretical construct, it can be emulated using existing computational methods and technologies. The main approaches to implementation include deterministic backtracking, annealing methods, particularly quantum annealing. Each of these methods has its own features and limitations, which are discussed below.

5.1. Deterministic Backtracking

One of the simplest ways to implement a machine with data transmission from the future is to use deterministic backtracking on a conventional computer. In this approach, the computation process is modeled as a tree of possible states, where each branch corresponds to one of the possible values of the symbol received from the future.

Working Principle:

(1) At each step of the computation, when reading a data symbol from the communication tape, the machine checks all possible values of that symbol.

(2) If the chosen value leads to a contradiction, the system returns to the decision point and tries the next value.

(3) The process continues until a consistent solution is found or all possible values are exhausted.

Advantages:

- Simplicity of implementation on existing computational platforms.
- Ability to use standard search algorithms with backtracking.

Disadvantages:

- High computational complexity, especially for problems with a large number of branches.
- Exponential growth of computation time as the depth of the decision tree increases.

5.2. Annealing Methods

Annealing methods represent a more complex approach to implementing a machine with time data transmission. In this case, the computation process is viewed as a whole structure (time unfolds in space). Then, a search is performed for the state of this structure with the lowest energy, where energy corresponds to the number of contradictions in the system.

Working Principle:

(*1*) Initially, the system is in an excited state with a high energy level, corresponding to the presence of many contradictions.

(*2*) Gradually, the system cools down, transitioning to states with lower energy.

(*3*) If a state with the lowest energy (without contradictions) is reached during the annealing process, the process is considered successful.

(*4*) If the lowest energy is achieved for only one answer (0 or 1), the result is considered final.

Advantages:

- Ability to find the global energy minimum even in complex systems with many local minima.
- Efficiency for problems where contradictions can be mitigated or eliminated during optimization.

Disadvantages:

- Requires significant resources to reach a state with minimal energy.
- Does not guarantee finding the optimal solution in polynomial time.

5.3. Quantum Annealing

Quantum annealing is a promising method for implementing a machine with data transmission from the future, based on the use of quantum devices. This method utilizes the effect of quantum tunneling to find states with the lowest energy.

Advantages:

- Potentially high speed of finding solutions due to quantum effects.
- Ability to solve problems that are difficult for classical computers.

Disadvantages:

- The speed and computational efficiency of quantum annealing are not yet fully understood.
- Existing quantum devices are limited in memory capacity and noise levels, complicating their use for complex tasks.

Prospects: Despite current limitations, quantum annealing remains a promising direction for experimentation. Existing quantum annealing machines, such as those from D-Wave (see, for example, [19]), are already suitable for research and can be used to explore the possibilities of implementing machines with data transmission from the future.

5.4. Summary

The implementation of a Turing machine with a time data transmission channel is possible using various approaches, including deterministic backtracking, annealing methods, and particularly quantum annealing. Each of these methods has its advantages and limitations, and the choice of a specific approach depends on the characteristics of the problem being solved and the available computational resources. Quantum annealing, in particular, remains a promising direction, although its practical efficiency requires further study.

6. Constructive Recognizers: An Example of Solving a Problem with a Time Data Transmission Machine

6.1. Non-constructiveness of Classical Recognizers

Classical recognition algorithms are usually non-constructive: they provide only a binary answer (YES or NO without explaining the result.

For example, when asked whether there exists a y that satisfies the condition $P(x, y)$, such algorithms merely confirm or deny the existence of such a y, without providing the actual object y upon a positive answer or proof of its absence upon a negative one. This limits their use in tasks where justification of the solution is required.

6.2. Constructive Problems in the Class NP ∩ co-NP

A Turing machine with a time data transmission channel allows the implementation of constructive recognizers that provide not only an answer but also a certificate—proof or witness of the result. We consider symmetric problems in the class NP ∩ co-NP, where the property $Q(x)$ is true if there exists a certificate y such that $Q_1(x, y)$, and false if there exists a certificate z such that $Q_0(x, z)$, with the sizes of y and z polynomially bounded relative to $|x|$.

6.3. Example: Checking for a Non-trivial Divisor

Consider the problem of determining whether the number x has a non-trivial divisor y such that $1 < y \leq d$ (the threshold d is given). This property $Q(d, x)$:

- If $Q(d, x) = 1$, then the certificate is the divisor y, verified by division in polynomial time.
- If $Q(d, x) = 0$, then the certificate is the factorization of x into prime factors, all $> d$, verified in polynomial time (for example, using the Miller test [20] under the assumption of the generalized Riemann hypothesis).

6.4. Implementation on a Time Data Transmission Machine

The constructive recognizer on a time data transmission machine works as follows:

(1) *Receiving Data from the Future*:
   - In the first step, the machine requests a pair (b, C) with label 0, where $b \in \{0, 1\}$ is the answer, and C is the certificate (divisor or factorization).
   - The answer b is immediately written to the output tape (if no fixed delay response is required, the certificate can also be issued, making the recognizer constructive).

(2) *Certificate Verification*:

- If $b = 1$, it checks that $C = y$ divides x and $1 < y \leq d$.
- If $b = 0$, it checks that C is a factorization of x into prime factors $> d$, and their product equals x.

(3) *Self-consistency*:

- After verification, if the certificate is correct, then (b, C) is written to the communication tape with label 0.
- If incorrect, a contradiction is created, and the computation branch is discarded.

It is easy to see that this machine operates in polynomial time.

Execution Example:

Input: $x = 15$, $d = 4$.

Step 1: Import $(1, 3)$ from the future, output 1.
Step 2: Check: $15 \div 3 = 5$, $1 < 3 \leq 4$, correct.
Step 3: Transmit $(1, 3)$ with label 0, self-consistency achieved.

Input: $x = 17$, $d = 4$.

Step 1: Import $(0, \{17\})$, output 0.
Step 2: Check: 17 is prime (Miller test), $17 > 4$, correct.
Step 3: Transmit $(0, \{17\})$, self-consistency achieved.

6.5. Features and Advantages

Linearity: The process is deterministic.

Instant Response: The answer is available in the early steps due to data transmission from the future.

Transparency: If a certificate is issued, it makes the result interpretable, enhancing the clarity of the computations.

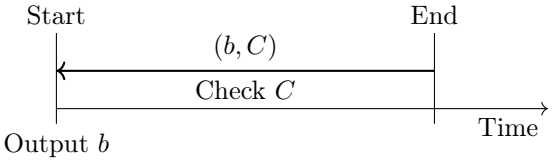

FIGURE 3. Scheme of the Constructive Recognizer's Operation

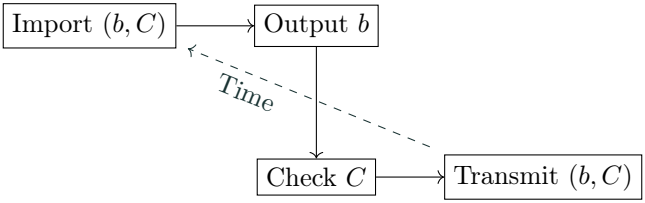

FIGURE 4. Flowchart of the Process

Figure 3 presents a scheme of the constructive recognizer's operation: data (b, C) is transmitted from the end of the computations to the beginning for immediate output and subsequent verification. The flowchart of the process from importing the result to verification and self-consistent data transmission is shown in Figure 4.

7. Actions for Time Data Transfer

To transform a conventional programming language into one that supports time data transfer, it is necessary to add two new operations: transferring data to the past and receiving data from the future. These actions allow the machine to interact with both future and past states of the computational process, providing new programming capabilities. In this section, we will examine these actions and their role in organizing computations with time data transfer.

7.1. Action of Transferring Data to the Past

The action of transferring data to the past allows the machine to record data at a specific moment in time, which can be used in previous computations. This action is formalized as follows:

Formal Description:

- *transfer*(l, d): transfers object d to the moment in time l.
  - l — the label of the moment in time for which the data is being transferred.
  - d — the object being transferred (e.g., bit, number, or string).

Working Principle:

(1) The machine records object d in the time data transfer channel with the timestamp l.

(2) The data becomes available for use in previous steps of computations related to the moment in time l.

(3) If a record with label l already exists, an error occurs.

Example: If the machine executes the action *transfer*$(1, 1)$, the value 1 is transferred with label 1. In previous computations related to 1, this value can be used for decision-making.

7.2. Action of Receiving Data from the Future

The action of receiving data from the future allows the machine to obtain data from a specific moment in the future, making it possible to use future information in current computations. This action is formalized as follows:

Formal Description:

- *accept*(l): accepts data from the moment in time with label l.
  - l — the name of the moment in time from which data is requested.

Working Principle:

(1) The machine requests data from the time data transfer channel with timestamp l.

(2) If the data exists, it is returned and can be used in current computations.

(3) If the data is absent, an error occurs.

Example: If the machine executes the action *accept*(1), it receives the value d, which was transferred with label 1 using the action *transfer*$(1, d)$. This value can then be used for decision-making in current computations.

7.3. Interaction of Data Transfers

The actions of transferring and receiving data work together, ensuring a connection between current and future states of the computational process. This interaction allows the machine to obtain data from the future for decision-making in current computations. It is important to note that the volume of data transferred through time is a significant indicator of the complexity of the computation. For example, when implementing a backtracking process, the runtime slows down exponentially with a binary indicator of the volume of data in bits, as for each bit of data, it effectively requires entering two branches of computation. Similar complexities arise in annealing methods.

7.4. Summary

The actions of transferring and receiving data through time are key elements of our computational model. In principle, we can incorporate these capabilities into any programming language, transforming it into a language for controlling a machine with data transfer from the future to the past. These tools provide new opportunities for creating intuitive computational systems.

8. Parallelization of the Computational Process

This section describes a parallel machine with time data transfer—an extension of our machine with an additional capability aimed at increasing its computational power: executing another instance of the machine with specific input data. The input string for the new process is placed on a special argument tape. The parallel process is initiated when the machine transitions to a special state for starting a parallel process. The new process begins with a special initial state of the subprocess. Initially, the new process runs until it reaches a state where it outputs a result (while the old process waits). This result is then transferred from the output tape of the new process to the argument tape of the old process. Both processes then continue to execute in parallel, with the old process no longer interacting with the new process. The parallel execution of the new process is necessary for transferring data to the past within it, ensuring the correctness of the result produced by this process.

The new process can be started as many times as necessary, either by the main process or by spawned subprocesses. Since subprocesses can launch their own subprocesses in parallel and independently, the number of subprocesses can grow exponentially as the machine operates in parallel mode.

The total parallel runtime of the machine is considered to be the time until all parallel processes complete, as if they were executed synchronously, step by step.

Theorem 3. *On a parallel machine with time data transfer, the operation of a conventional Turing machine with polynomial memory can be modeled in polynomial time (which corresponds to the complexity class* PSPACE*).*

Proof. Consider a Turing machine with a known estimate of its capacity complexity in the class PSPACE. We will perform the following steps:

(1) First, estimate the runtime of this machine (exponential in the volume of memory).
(2) Launch a subprocess to model the operation of the machine for half of this time (the modeling time is passed to the subprocess along with the initial configuration of the process).
(3) After obtaining the result and allowing the subprocess to continue forming it in the future, immediately launch a new subprocess to model the operation of the original machine for the remaining half of the allocated time, passing this subprocess the configuration obtained from the first subprocess.
(4) Output the result.
(5) The subprocesses continue working in a similar manner until the allocated time converges to a predetermined constant.

It is easy to see that the depth of subprocess calls here is polynomial, and the total parallel runtime is also polynomial. This completes the proof of the theorem. □

Theorem 4. *The operation of a parallel machine (with time data transfer) in polynomial time can be modeled by a conventional Turing machine with polynomial memory.*

PROOF. The reverse statement follows from the fact that the operation of such a parallel machine can be modeled on a conventional Turing machine using polynomial memory. This is achieved by sequentially executing all launched processes, as they do not interact with each other. Since the total parallel time of the modeled machine is polynomially bounded, the depth of subprocess calls is also polynomially bounded. Thus, the total memory volume for modeling also remains polynomial. This proves the theorem. □

9. Direct Access to Input Data

In this section, we modify the model of our machine by adding the capability to work with large data through direct access to the cells of the input tape. By large data, we mean data that cannot be sequentially examined within the current time constraints. The direct access mechanism allows for quick access to remote cells, overcoming the specified limitation. This is achieved as follows: a special address tape is introduced, on which the binary representation of the cell number on the input tape is recorded. The input tape head instantly moves to the specified cell. No other mechanisms for moving this head are provided. We assume that all launched subprocesses work with the same input data. Thus, we have defined a parallel machine with time data transfer and direct access to input data.

If we replace *PSPACE* with *LOGSPACE* in the previous section, we obtain the following theorem.

THEOREM 5. *On a parallel machine with time data transfer and direct access to input data, the operation of a conventional Turing machine with logarithmic memory (LOGSPACE) can be modeled in logarithmic time (LOGTIME).*

PROOF. It is similar to the proof of Theorem 3, but with the replacement of polynomial constraints with logarithmic ones. Since access to input data occurs in constant time, the total execution time remains logarithmic. □

The reverse statement is problematic since direct modeling of logarithmic time on our machine using a conventional Turing machine requires a memory volume polynomial in the logarithm of the size of the original data. However, a similar approach can be used to prove the following theorem.

Theorem 6. *The class POLYLOGTIME on a parallel machine with time data transfer and direct access to input data coincides with the class POLYLOGSPACE for conventional Turing machines.*

Proof. The proof is based on the fact that the operation of our machine with direct access can be modeled on a conventional Turing machine using polylogarithmic memory. Thus, the total modeling time remains polylogarithmic. □

Conclusion

This work presents a fundamentally new computational model that expands classical representations of computational processes through the mechanism of transferring data across different moments in time. The main results of the research are as follows:

(1) *Complexity Classes and Time Constraints*: It has been proven that a machine with data transfer from the future, without time constraints, recognizes exactly the decidable sets with a constant response delay. Under polynomial time constraints, the model corresponds to the class of problems in the intersection of *NP* and *co-NP*. These results demonstrate how controlled violations of causal order can assist computations.

(2) *Parallelization and Space Complexity*: The extension of the model to allow the launching of subprocesses that inherit input data enables the coverage of the *PSPACE* class in polynomial time, demonstrating how time complexity transforms into space complexity.

(3) *Direct Access and Data Processing Efficiency*: The modification with addressable access to the input tape and logarithmic time complexity ensures compliance with the *LOGSPACE* class, which is particularly relevant for large data processing tasks. This result is supported by the mechanism of instant movement of the input tape head, eliminating linear delays.

(4) *Practical Implementation*: Two approaches to the practical implementation of the model are proposed:

- *Deterministic Backtracking* — a systematic search for self-consistent computational trajectories;

- *Annealing Methods* (including quantum systems) — searching for states with minimal energy of contradictions.

Experiments with quantum annealers, such as D-Wave, could be the next step in assessing the physical realizability of the model.

The prospects for further research encompass both theoretical and practical aspects:

- Optimization of the volume of transmitted data to reduce computational load;
- Development of specialized programming languages with operators for time data transfer;
- Analysis of the connection with quantum computing.

This work is relevant not only for theoretical computer science but also for interdisciplinary research. It demonstrates that the connection of states in time may be the key to solving problems inaccessible to traditional systems. The proposed model could serve as a bridge between complexity theory and physical realizations, opening new possibilities for algorithms.

References

[1] S. A. Cook. "The complexity of theorem-proving procedures", *Logic, automata, and computational complexity: The works of Stephen A. Cook*, ed. Kapron B.C., ACM, New York, 2023, ISBN 979-8-4007-0779-7, pp. 143–152. ↑5

[2] R. M. Karp. "Reducibility among combinatorial problems", *50 Years of Integer Programming 1958–2008. From the Early Years to the State-of-the-Art*, eds. Jünger M., et al., Springer, Berlin–Heidelberg, 2010, ISBN 978-3-540-68274-5, pp. 219–241. ↑5

[3] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Series of Books in the Mathematical Sciences, Freeman, San Francisco, 1979, ISBN 978-0716710455, 340 pp. ↑5

[4] S. Aaronson. "Guest column: NP-complete problems and physical reality", *ACM Sigact News*, **36**:1 (2005), pp. 30–52. ↑6

[5] D. Deutsch. "Quantum theory, the Church–Turing principle and the universal quantum computer", *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **400**:1818 (1985), pp. 97–117. ↑6

[6] T. A. Brun. "Computers with closed timelike curves can solve hard problems efficiently", *Foundations of Physics Letters*, **16**:3 (2003), pp. 245–253. ↑6

[7] S. Aaronson, J. Watrous. "Closed timelike curves make quantum and classical computing equivalent", *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **465**:2102 (2009), pp. 631–647. ↑6

[8] Ch. Papadimitriou. *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994, ISBN 978-0201530827, 523 pp. ↑6

[9] S. Arora, B. Barak. *Computational Complexity: A Modern Approach*, Cambridge University Press, New York, 2009, ISBN 978-0521424264, 594 pp. ↑6

[10] M. A. Nielsen, I. L. Chuang. *Quantum computation and quantum information*, Cambridge University Press, New York, 2010, ISBN 9781107002173, 702 pp. ↑6

[11] L. K. Grover. "A fast quantum mechanical algorithm for database search", *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (22–24 May 1996, Pennsylvania, USA), ACM, New York, 1996, ISBN 978-0-89791-785-8, pp. 212–219. ↑6

[12] P. W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring", *Proceedings 35th Annual Symposium on Foundations of Computer Science* (20–22 November 1994, Santa Fe, NM, USA), IEEE, 1994, ISBN 0-8186-6580-7, pp. 124–134. ↑6

[13] A. W. Harrow, A. Montanaro. "Quantum computational supremacy", *Nature*, **549**:7671 (2017), pp. 203–209. ↑6

[14] K. Iqbal, O. Ahmad, A. Y. Vandika. "Quantum computing and its implications for complex system analysis", *Research of Scientia Naturalis*, **1**:5 (2024), pp. 238–247. ↑6

[15] J. I. Orlicki. *Time-travelling Turing Machines and the self-consistent Halting Problem*, 2024. ↑6

[16] K. Gödel. "An example of a new type of cosmological solutions of Einstein's field equations of gravitation", *Rev. Mod. Phys.*, **21**:3 (1949), pp. 447–450. ↑7

[17] D. Bacon. "Quantum computational complexity in the presence of closed timelike curves", *Phys. Rev. A*, **70**:3 (2004), id. 032309. ↑7

[18] D. H. Wolpert. "Physical limits of inference", *Physica D: Nonlinear Phenomena*, **237**:9 (2008), pp. 1257–1281. ↑7

[19] A. D. King, J. Raymond, T. Lanting, R. Harris, A. Zucca, F. Altomare, A. J. Berkley, K. Boothby, S. Ejtemaee, C. Enderud, E. Hoskinson, Sh. Huang, E. Ladizinsky, A. J. R. MacDonald, G. Marsden, R. Molavi, T. Oh, G. Poulin-Lamarre, M. Reis, Ch. Rich, Y. Sato, N. Tsai, M. Volkmann, J. D. Whittaker, J. Yao, A. W. Sandvik, M. H. Amin. "Quantum critical dynamics in a 5,000-qubit programmable spin glass", *Nature*, **617**:7959 (2023), pp. 61–66. ↑7, 17

[20] G. L. Miller. "Riemann's hypothesis and tests for primality", *STOC '75: Proceedings of the seventh annual ACM symposium on Theory of computing* (5–7 May 1975, Albuquerque, New Mexico, USA), ACM, New York, ISBN 978-1-4503-7419-4, pp. 234–239. doi ↑18

Received 15.03.2025;
approved after reviewing 02.04.2025;
accepted for publication 02.04.2025;
published online 21.04.2025.

Information about the authors:

Milad Joudakizadeh

PhD student in Artificial Intelligence and Machine Learning, Research focuses on computational complexity, machine learning, and logic in computer science.

0000-0002-6167-6237
e-mail: joudakizadeh@mail.ru

Anatoly Petrovich Beltiukov

Doctor of Physics and Mathematics, Professor. Research focuses on computational complexity, complexity classes, subrecursive hierarchies, intuitionistic mathematics, constructive systems, proof mechanization, proof complexity, computation models, substructural logics, weak arithmetics, and logic in computer science.

0000-0002-3433-9067
e-mail: belt.udsu@mail.ru

The authors contributed equally to this article.

The authors declare no conflicts of interests.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ТЕОРЕТИЧЕСКИЕ ОСНОВАНИЯ ПРОГРАММНЫХ СИСТЕМ

УДК 004.415.2:519.681
10.25209/2079-3316-2025-16-2-3-54

Сложность вычислений с путешествиями во времени

Милад **Джудакизаде**[1], Анатолий Петрович **Бельтюков**[2]✉
[1,2] Удмуртский государственный университет, Ижевск, Россия
[2]✉ *belt.udsu@mail.ru*

Аннотация. Эта работа рассматривает математическую модель вычислений, которая может интерпретироваться как компьютер, способный получать данные из будущих состояний своего вычислительного процесса. При определённых сочетаниях входных данных и программ такие вычисления могут становиться невыполнимыми из-за возникающих противоречий или приводить к неоднозначным результатам. Мы исследуем программы, для которых этот процесс всегда выполним и даёт однозначный результат.

Показано, что при отсутствии ограничений на вычислительную сложность такие машины могут выдавать значение любого рекурсивно разрешимого предиката через фиксированное время после начала вычисления – время задержки ответа (при этом процесс вычисления должен продолжаться и после получения ответа). Если общее время работы таких машин ограничить полиномами от размера входных данных, то эти машины будут распознавать в точности языки, принадлежащие пересечению классов NP и co-NP, за постоянное время задержки ответа в таком же смысле.

Рассматриваются возможные реализации такого компьютера на практике, включая анализ возможных протоколов работы с использованием квантового отжига для выбора нужного процесса. Показано, что при распараллеливании вычислительного процесса класс задач, решаемых рассматриваемыми машинами за полиномиальное время, соответствует классу PSPACE.

Кроме того, исследуется режим работы, при котором эти машины имеют прямой доступ ко входным данным. В этом случае, если время работы ограничено логарифмом от размера входных данных, то класс задач, решаемых таким параллельно работающим компьютером, содержит LOGSPACE.

Результаты данного исследования могут быть использованы для разработки новых принципов программирования недетерминированных вычислительных машин, в которых вместо недетерминированных выборов используется передача данных из будущего.

(*Связанные тексты статьи на английском и на русском языках*)

Ключевые слова и фразы: Машина Тьюринга, вычислительная сложность, рекурсивные множества, недетерминированные вычисления, квантовый отжиг, машина времени, искусственный интеллект

Для цитирования: Джудакизаде М., Бельтюков А. П. *Сложность вычислений с путешествиями во времени* // Программные системы: теория и приложения. 2025. **Т. 16**. № 2(65). С. 3–54. (*Англ.+русс.*) https://psta.psiras.ru/read/psta2025_2_3-54.pdf

Введение

В данной статье мы рассматриваем необычный взгляд на вычислительные процессы. Что получается, если машине разрешать заглядывать в своё собственное будущее и использовать эту информацию для принятия решений в настоящем? Эта идея, которая на первый взгляд кажется внутренне противоречивой, лежит в основе исследуемой вычислительной модели — машины Тьюринга с каналом передачи данных против течения времени. Эта модель не только переосмысливает понимание времени в вычислениях, но и поднимает фундаментальные вопросы о причинности и взаимодействии между прошлым и будущим в вычислительных процессах.

Наша машина определена таким образом, что она может получать информацию из будущих состояний своего вычислительного процесса. Эта модель представляет собой теоретическую конструкцию, вводящую новые способы организации вычислений. Она позволяет получить результат сложного вычисления до его завершения и использовать эту информацию. Как и в любом путешествии во времени, здесь возникают сложности: передача данных из будущего может привести к противоречиям и неоднозначностям.

В данной статье мы сосредотачиваемся на случаях, когда таких проблем можно избежать, и демонстрируем, как эту модель можно реализовать и применить для решения сложных задач. Этот подход представляет собой альтернативную вычислительную парадигму, которая связывает время, причинность и вычисления, позволяя применять новые стратегии решения задач.

Одним из результатов является то, что такая машина может вычислить значение любого рекурсивно разрешимого предиката за постоянное время. Это означает, что задачи, требующие значительного времени в классических моделях, могут быть решены практически мгновенно (хотя сам вычислительный процесс должен продолжаться и общее его время может оказаться весьма значительным).

Мы доказываем также, что если общее время работы таких машин ограничивать полиномами от размера входных данных, то эти машины смогут распознавать все языки из пересечения классов NP и co-NP. Эти результаты говорят о том, что предложенная модель не только теоретически интересна, но и обладает значительной вычислительной мощностью.

Мы также рассматриваем модификацию машины, которая позволяет осуществлять параллельную обработку данных. Показано, что в этом случае класс задач, решаемых такими машинами за полиномиальное

время, соответствует PSPACE (то есть полиномиальное время на новых машинах эквивалентно PSPACE на обычных машинах Тьюринга).

Другая модификация предполагает добавление механизма прямого доступа ко входным данным через адресацию ячеек входной ленты с помощью специальной адресной ленты. В этом случае, при ограничении времени работы логарифмом от размера входных данных, класс задач, решаемых нашим распараллеленным компьютером, включает LOGSPACE (здесь LOGTIME также преобразуется в соответствующую емкостную сложность). Стоит отметить, что такой режим работы не влечёт чрезмерно высокой сложности и является реализуемым на обычных компьютерах при разумных размерах входных данных.

С практической точки зрения, наша работа вводит новые механизмы программирования. Вместо сложных недетерминированных выборов, которые могут быть трудны для понимания, мы предлагаем использовать данные из будущего, что делает процесс вычислений интуитивно более понятным. Мы также исследуем возможные реализации этой модели, включая детерминированный поиск с возвратом и квантовый отжиг.

В первом подходе систематический поиск с возвратами назад гарантирует, что все возможные пути будут исследованы для предотвращения парадоксов. Во втором подходе квантовые адиабатические машины одновременно исследуют множество состояний вычислительного процесса, выбирая самосогласованную последовательность действий.

Таким образом, мы предлагаем модель, которая расширяет наше понимание вычислительных процессов и предоставляет новые подходы к программированию.

1. Связанные работы

За последние несколько десятилетий идея использования временных парадоксов и передачи данных во времени привлекла внимание в теоретической информатике и физике. Многие исследователи изучали возможность расширения классических вычислительных моделей за счёт включения механизмов, позволяющих получать информацию из будущего, что, как предполагается, может переопределить традиционные взгляды на вычислительные ресурсы и сложность.

В основополагающей работе Кука [1] было введено понятие NP-полноты, заложившее основу для дальнейших исследований в области вычислительной сложности и сводимости различных задач к универсальной проблеме. Развивая эти идеи, Р. М. Карп [2] проанализировал сводимость комбинаторных задач, показав, что многие на первый взгляд различные задачи взаимосвязаны.

Классическая работа Гэри и Джонсона [3] стала фундаментальным справочником для исследователей, изучающих вычислительную сложность, представив широкий спектр NP-полных задач и послужив отправной точкой для анализа ограничений традиционных вычислительных моделей.

В последние годы значительное внимание уделяется моделям, использующим физические эффекты для решения сложных вычислительных задач. Ааронсон [4] исследовал потенциал квантовых вычислений для решения традиционно сложных задач, продемонстрировав, что квантовые методы могут предоставить существенные преимущества перед классическими алгоритмами.

Дойч [5] ввёл концепцию универсального квантового компьютера, что стало важной вехой в квантовых вычислениях и повлияло на дальнейшие исследования гибридных вычислительных моделей.

Что касается моделей, нарушающих классическую причинность, существенный вклад был сделан в исследованиях, связанных с замкнутыми времениподобными кривыми (ЗВК). Брун [6] показал, что использование ЗВК может значительно ускорить решение задач. Кроме того, Ааронсон и Уотрус [7] установили эквивалентность квантовых и классических вычислений в присутствии ЗВК, подчеркнув потенциал таких моделей для расширения вычислительных возможностей.

Помимо классических подходов, идея интеграции передачи данных на основе путешествий во времени с концепциями теории сложности вызвала интерес. Работы Пападимитриу [8] и Ароры и Барака [9] предоставляют глубокий анализ структуры вычислительных задач, включая связь между классами NP и co-NP. Эти исследования служат основой для анализа того, как нестандартные вычислительные модели, использующие передачу данных из будущего, могут решать задачи, с которыми традиционные модели справляются с трудом.

Классические работы по квантовым вычислениям, такие как книга Нильсена и Чанга [10], а также алгоритмы Гровера [11] и Шора [12], иллюстрируют, как квантовые методы могут значительно ускорить решение определённых классов задач.

Более поздние исследования, такие как работы Харроу и Монтанаро [13] и исследования Икбала и др. [14], подтверждают практическую значимость квантовых алгоритмов для решения сложных вычислительных задач.

Важным аспектом этой области является концепция передачи данных на основе путешествий во времени, позволяющая машинам предсказывать свои будущие состояния и использовать эту информацию для корректировки текущих вычислений. В этом контексте работа Орлицки [15] исследует

модель машин с путешествиями во времени и рассматривает проблему самосогласованной остановки. Этот подход расширяет традиционные модели, предлагая новые взгляды на взаимодействие временных потоков в вычислениях.

Кроме того, вклад Гёделя [16] продемонстрировал теоретическую возможность замкнутых времениподобных кривых в космологических решениях, предоставив дополнительное физическое обоснование для вычислительных моделей, включающих передачу данных из будущего.

Исследования Бэкона [17] и Вольперта [18] указывают на то, что физические ограничения на предсказание будущего и обратную связь между временными состояниями значительно влияют на вычислительные процессы.

Наконец, экспериментальные исследования, такие как работа Кинга и др. [19], показывают, что квантовые системы с тысячами кубитов уже могут моделировать сложные физические явления, что вселяет надежды на практическую реализуемость вычислительных моделей, использующих эффекты путешествий во времени.

Таким образом, эти работы иллюстрируют, что идея передачи данных из будущего и эффектов временных петель в вычислениях имеет прочные теоретические и практические основания. Наша модель вычислений с путешествиями во времени естественным образом вписывается в этот контекст, предлагая новый взгляд на вычислительную сложность и позволяя решать задачи, недоступные для традиционных моделей.

2. Структура и особенности машины Тьюринга с каналом передачи данных во времени

В данном разделе описывается конструкция машины Тьюринга, оснащённой каналом передачи данных во времени. Эта модель расширяет классическую машину Тьюринга за счёт включения механизма, позволяющего передавать данные на ранее выполненные шаги вычислительного процесса и получать данные, переданные из будущих шагов. Рассмотрим структуру и особенности такой машины.

2.1. Основные компоненты машины

Машина Тьюринга с каналом передачи данных во времени является многоленточной машиной (с одной головкой на каждую ленту) и включает следующие элементы:

Входная лента: На этой ленте содержится входное слово, которое необходимо обработать. В начале работы машины входные данные записываются на эту ленту.

Выходная лента: На этой ленте записывается результат работы машины. Для распознающей машины результат может быть представлен в виде бинарного значения: 0 (нет) или 1 (да).

Несколько рабочих лент для выполнения необходимых вычислений.

Коммуникационная лента (лента связи): Специальная лента, предназначенная для передачи данных во времени. Машина может записывать данные на эту ленту для отправки в прошлое и получать данные из будущего. Для отправки и получения данных машина должна перейти в специальные состояния передачи и приёма данных соответственно.

Адресная лента для передаваемых данных: На этой ленте записываются метки, которые позволяют извлекать данные, ссылаясь на эти метки в запросах. Для передачи и извлечения данных используются специальные состояния машины.

В состоянии передачи содержимое коммуникационной ленты вместе с меткой отправляется в канал передачи данных во времени. Если данные с такой меткой уже были ранее отправлены, то считаем, что при работе машины возникла ошибка.

В состоянии приёма коммуникационная лента получает данные из канала передачи, связанные с заданной меткой. Если таких данных в будущем не оказалось, то также считаем, что при работе машины возникла ошибка. Для решения задач распознавания машины, при работе которых могут возникать ошибки, использовать запретим.

Алфавиты лент: Каждая лента имеет свой алфавит символов, которые могут быть записаны на неё.

Управляющее устройство: На каждом шаге работы управляющее устройство может находиться в одном из состояний алфавита состояний. Выделяются следующие специальные состояния:

начальное состояние, в котором машина начинает вычислительный процесс;

заключительное состояние, в котором работа машины завершается;

состояние передачи данных, при переходе в которое данные передаются с указанной меткой в канал передачи данных;

состояние приёма данных, при переходе в которое данные принимаются из канала передачи данных с указанной меткой;

состояние вывода, при переходе в которое содержимое выходной ленты отправляется во внешний мир, пока машина продолжает свою работу; считаем, что машина работает с однократной записью на выходную ленту, то есть повторный переход в состояние вывода — ошибка в работе машины.

Программа: Функция (таблица), которая отображает состояние машины и символы, видимые головками на каждой ленте, в новое состояние, новые символы, записываемые на ленты, и направления сдвигов головок.

Время вычислений: Последовательность шагов. Каждый шаг выполняет одну команду машины или передаёт/принимает один символ через канал передачи данных во времени (то есть на передачу длинного слова в канал передачи данных тратится столько шагов, какова длина этого слова). Для распознающей машины время вычислений должно быть конечным.

На диаграмме рисунка 1 показан процесс обработки данных с возможностью передачи информации из будущего в прошлое через временной сдвиг, реализуемый замкнутыми временеподобными кривыми в метрике Керра.

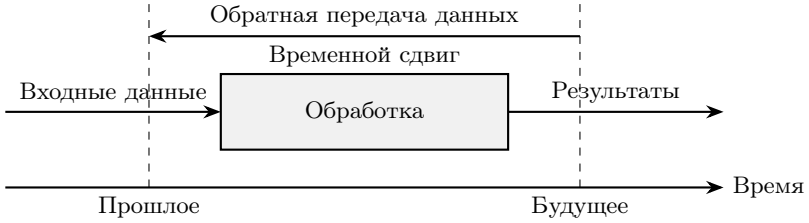

Рисунок 1. Схема работы машины Тьюринга с обратной временной связью

2.2. Особенности работы машины

Работа машины Тьюринга с каналом передачи данных во времени имеет несколько отличительных особенностей:

Вывод результата до завершения вычислений: Машина может вывести результат на выходную ленту до завершения процесса вычислений. Это приводит к разделению двух понятий времени:

время задержки ответа,
общее время вычислений.

Однозначность и определённость вычислений: Из-за возможности получения данных из будущего работа машины может стать неоднозначной или неопределённой. Например, неоднозначность возникает, если машина, получив некоторые данные из будущего, записывает те же данные с той же меткой, что может привести к недетерминированности процесса и, как следствие, к различным конечным результатам, в зависимости от того, какие данные окажутся прочитанными и записанными. Если машина, прочитав данные с одной меткой, затем обязательно записывает отличные от прочитанных данные с той же меткой, то возникает противоречие, делающее процесс невозможным, что приводит к неопределённости. Машина считается

распознающей некоторое свойство входных данных (т.е. вычисляющей характеристическую функцию некоторого множества), если её работа определена для всех входных данных и результат однозначен.

Программируемость: В конце статьи мы опишем инструменты алгоритмических языков, которые позволят программировать подобные действия не только для машин Тьюринга, но и для других компьютеров.

2.3. Временные аспекты работы машины

Ключевой особенностью машины является возможность получения данных из будущего, что приводит к следующим временным эффектам:

Постоянное время задержки ответа: Благодаря механизму передачи данных во времени машина может вывести простой результат (ДА или НЕТ) за постоянное время, независимо от вычислительной сложности.

Общее время вычислений: Несмотря на постоянное время задержки ответа, общее время вычислений может быть значительно больше, так как вычислительный процесс должен продолжаться даже после вывода результата.

2.4. Объяснение неоднозначности и неопределённости

Чтобы проиллюстрировать неоднозначность и неопределённость в работе машины, рассмотрим следующий пример. Предположим, что машина запрашивает один бит из будущего и выбирает ветвь вычислений на основе его значения. Если получено значение бита 1, то машина следует по первой ветви; если получено значение 0, то она следует по второй ветви. Неоднозначность процесса вычисления возникает, если в каждой ветви машина записывает тот же бит с той же меткой на коммуникационную ленту. Это создаёт два возможных пути вычисления. Следует отличать неоднозначность вычисления от неоднозначности результата. При неоднозначном процессе вычисления результат может оказаться одним и тем же.

Если машина считывает один бит из будущего и пытается записать противоречащий ему бит на коммуникационную ленту с той же меткой, то возникает противоречие, приводящее к невозможности такого вычисления.

Комбинации этих ситуаций могут приводить к тому, что некоторые ветви процесса вычисления, возникшие из-за внутренней неоднозначности, затем могут оказаться противоречивыми и быть по этой причине отсеяны. Тогда останутся только непротиворечивые ветви. Может оказаться так, что все оставшиеся непротиворечивые ветви приводят к одному окончательному результату. Если всегда происходит так, то будем считать такую машину корректной для решения задачи распознавания.

Таким образом, описанные ситуации могут привести к следующему:

к неоднозначности, если несколько ветвей вычислений завершаются разными результатами,

к неопределённости, если ни одна ветвь не может завершиться без противоречий.

Это демонстрирует, как использование данных из будущего может повлиять на вычислительный процесс.

3. Возможности машин с передачей данных во времени

Схема передачи данных во времени в машине Тьюринга с замкнутыми времениподобными кривыми представлена на рисунке 2. Показаны основные этапы обработки информации, механизм обратной связи через временные петли и направление передачи данных.

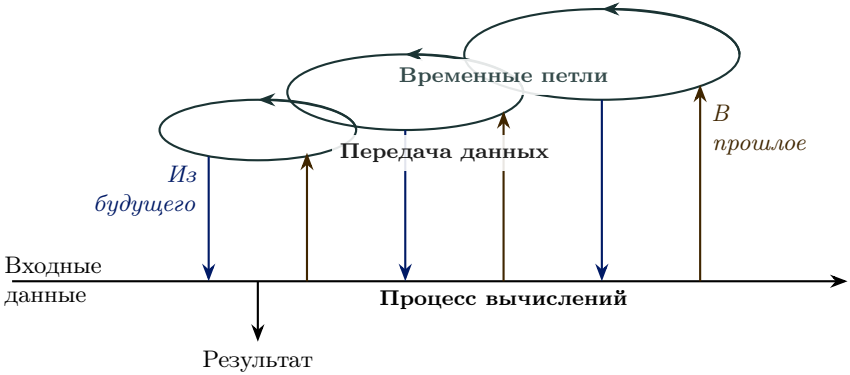

Рисунок 2. Схема временной передачи данных в машине Тьюринга с замкнутыми времениподобными кривыми

В данном разделе рассматриваются вычислительные возможности машины Тьюринга с каналом передачи данных во времени. Мы представляем две теоремы, которые демонстрируют, как использование данных из будущего влияет на вычислительную сложность и классы задач, решаемых такой машиной. Эти теоремы показывают, что предложенная модель обладает свойствами, позволяющими решать задачи, недоступные для классических вычислительных систем.

3.1. Распознавание разрешимых множеств

Теорема 1. *Машина Тьюринга с каналом передачи данных во времени распознаёт с постоянным временем задержки ответа в точности все рекурсивно разрешимые множества.*

Доказательство. Доказательство основывается на способности машины использовать данные из будущего для предсказания результатов вычислений. Пусть нам надо распознать принадлежность входных слов к заданному рекурсивно разрешимому множеству. Для этого множества имеется обычная распознающая машина Тьюринга. Из этой машины построим требуемую машину Тьюринга с каналом передачи данных во времени, которая будет работать следующим образом.

В начале работы машина запрашивает содержимое коммуникационной ленты с меткой 0. Это значение и выдаётся на выход в качестве результата работы. Затем запускается процесс работы обычной машины Тьюринга, распознающей принадлежность заданному множеству. Когда распознавание закончится, его результат передаётся в прошлое с меткой 0. Таким образом обеспечивается, что в прошлом машина выдала верный результат.

Теперь пусть мы имеем исходную машину с каналом передачи данных по времени, распознающую принадлежность входных слов некоторому множеству. По определению, общее время работы такой машины всегда конечно. Построим новую обычную машину Тьюринга, распознающую принадлежность этому множеству, которая перебирает все возможные слова, претендующие на роль частичного протокола работы исходной машины (цепочки мгновенных описаний), в порядке их удлинения.

Машина проверяет для каждого слова, является ли оно правильным завершённыи протоколом работы исходной машины. Как только такой протокол найден, результат работы исходной машины, выдаётся в качестве результата новой машины. Поскольку процесс работы исходной машины, по определению, определён и однозначен, новая машина всегда выдаёт правильный ответ о принадлежности входного слова заданному множеству. □

3.2. Распознавание множеств в NP ∩ co-NP

Теорема 2. *Машина Тьюринга с постоянным временем задержки ответа и полиномиальным общим временем вычислений распознаёт в точности все множества в* NP ∩ co-NP.

Идея доказательства. интерпретируя приём данных из будущего как ветвление на несколько путей, мы строим машину, которая работает за полиномиальное время и обеспечивает согласованность результатов на всех ветвях вычислений.

4. Доказательство теоремы 2

В данном разделе мы изложим доказательство теоремы 2, сформулированной в предыдущем разделе. Сначала мы переформулируем работу нашей машины в терминах недетерминированных вычислений. Это позволит нам преобразовать работу нашей машины в работу обычных недетерминированных машин Тьюринга и наоборот. Поскольку механизм перехода к постоянному времени задержки описан при доказательстве предыдущей теоремы, то для начала достаточно доказать теорему 2 без упоминания об этом, что мы и сделаем.

4.1. Переход к недетерминированным вычислениям

Идея заключается в том, чтобы интерпретировать чтение каждого символа данных на коммуникационной ленте, полученного из будущего по некоторой метке, как ветвление процесса вычислений. Каждая ветвь соответствует одному из возможных значений этого символа. Когда машина достигает момента времени, обозначенного данной меткой, она проверяет, совпал ли предполагаемый считанный символ с переданным. Если значение символа, записанного в ячейке коммуникационной ленты, совпадает с ожидаемым значением в текущей ветви, вычисления продолжаются. В противном случае ветвь вычислений завершается из-за возникшего противоречия.

Корректная распознающая машина должна давать одинаковые ответы (либо 0 — НЕТ, либо 1 — ДА) на всех непротиворечиво завершающихся ветвях для данного входного слова. Далее мы преобразуем нашу исходную машину с передачей данных во времени в недетерминированную, как показано выше, а затем разделим её на две недетерминированные машины Тьюринга, обработав, упомянутые выше противоречия следующим образом.

(1) Первая машина принимает распознаваемое множество, возвращая 1, когда исходная машина выводит 1, а во всех остальных случаях (включая противоречия) возвращает 0.

(2) Вторая машина принимает дополнение этого множества, возвращая 1, когда исходная машина выводит 0, а во всех остальных случаях (включая противоречия) возвращает 0.

Таким образом, если исходная машина с передачей данных во времени выдаёт да, то первая из полученных машин на некоторых ветвях вычисления выдаёт ДА, а вторая — всегда НЕТ. Наоборот, если исходная машина выдаёт НЕТ, то первая всегда выдаёт НЕТ, а вторая — на некоторых ветвях вычисления выдаёт ДА. Таким образом, первая из построенных детерминированных принимает само распознаваемое множество, а вторая — его дополнение. Нетрудно видеть, что полиномиальное ограничение времени при указанном преобразовании сохраняется, так как все проведённые перестройки дают не более, чем полиномиальное замедление. Так что, если исходная машина распознавала данное множество за полиномиальное время, то такое же время работают и вновь построенные машины. Это означает, что исходно распознаваемое множество принадлежит $\mathsf{NP} \cap \mathsf{co\text{-}NP}$.

4.2. Построение машины с передачей данных во времени из недетерминированных машин

Обратное утверждение также верно: если существуют две недетерминированные машины, ограниченные полиномиальным временем, одна из которых принимает некоторое множество, а другая принимает его дополнение, мы можем из них построить одну машину с передачей данных из будущего, распознающую это множество и работающую полиномиальное время. Эта машина работает следующим образом:

(1) В начале процесса она угадывает, какую из двух недетерминированных машин запустить.

(2) Затем она выполняет выбранную машину.

(3) Если первая выбранная машина выводит 1, ветвь вычислений завершается с результатом 1.

(4) Если первая выбранная машина выводит 0, ветвь вычислений приводится к противоречию.

(5) Если вторая выбранная машина выводит 1, ветвь вычислений завершается с результатом 0.

(6) Если вторая выбранная машина выводит 0, ветвь вычислений приводится к противоречию.

По условию, ровно одна из этих машин должна дать положительный результат. Таким образом, наша машина всегда даёт определённый ответ: либо 1, либо 0.

4.3. Постоянное время задержки ответа

Постоянное время задержки ответа достигается благодаря тому, что решение о результате вычислений принимается в самом начале процесса. Машина запрашивает данные из будущего и использует их для немедленного вывода ответа. При этом процесс вычислений должен продолжаться. □

5. Вопросы возможной реализации

В данном разделе обсуждаются возможные подходы к практической реализации машины Тьюринга с каналом передачи данных во времени. Хотя такая машина является теоретической конструкцией, её можно эмулировать с использованием существующих вычислительных методов и технологий. Основные подходы к реализации включают детерминированный поиск с возвратом, методы отжига, в частности, квантовый отжиг. Каждый из этих методов имеет свои особенности и ограничения, которые обсуждаются ниже.

5.1. Детерминированный поиск с возвратом

Один из самых простых способов реализации машины с передачей данных из будущего заключается в использовании детерминированного поиска с возвратом на обычном компьютере. В этом подходе процесс вычислений моделируется как дерево возможных состояний, где каждая ветвь соответствует одному из возможных значений символа, полученного из будущего.

Принцип работы:

(1) На каждом шаге вычислений, при чтении символа данных с коммуникационной ленты, машина проверяет все возможные значения этого символа.

(2) Если выбранное значение приводит к противоречию, система возвращается к точке принятия решения и пробует следующее значение.

(3) Процесс продолжается до тех пор, пока не будет найдено согласованное решение или не будут исчерпаны все возможные значения.

Преимущества:

- Простота реализации на существующих вычислительных платформах.
- Возможность использования стандартных алгоритмов поиска с возвратом.

Недостатки:

- Высокая вычислительная сложность, особенно для задач с большим количеством ветвлений.
- Экспоненциальный рост времени вычислений с увеличением глубины дерева решений.

5.2. Методы отжига

Методы отжига представляют собой более сложный подход к реализации машины с передачей данных во времени. В этом случае процесс вычислений рассматривается как целостная структура (время разворачивается в пространстве). Затем выполняется поиск состояния этой структуры с наименьшей энергией, где энергия соответствует количеству противоречий в системе.

Принцип работы:

(1) Изначально система находится в возбуждённом состоянии с высоким уровнем энергии, что соответствует наличию множества противоречий.
(2) Постепенно система охлаждается, переходя в состояния с меньшей энергией.
(3) Если в процессе отжига достигается состояние с наименьшей энергией (без противоречий), процесс считается успешным.
(4) Если наименьшая энергия достигается только для одного ответа (0 или 1), результат считается окончательным.

Преимущества:

- Возможность нахождения глобального минимума энергии даже в сложных системах с множеством локальных минимумов.
- Эффективность для задач, где противоречия могут быть смягчены или устранены в процессе оптимизации.

Недостатки:

- Требует значительных ресурсов для достижения состояния с минимальной энергией.
- Не гарантирует нахождение оптимального решения за полиномиальное время.

5.3. Квантовый отжиг

Квантовый отжиг является перспективным методом реализации машины с передачей данных из будущего, основанным на использовании квантовых устройств. Этот метод использует эффект квантового туннелирования для нахождения состояний с наименьшей энергией.

Преимущества:

- Потенциально высокая скорость нахождения решений благодаря квантовым эффектам.
- Возможность решения задач, сложных для классических компьютеров.

Недостатки:

- Скорость и вычислительная эффективность квантового отжига ещё не до конца изучены.
- Существующие квантовые устройства ограничены по объёму памяти и уровню шума, что осложняет их использование для сложных задач.

Перспективы: Несмотря на текущие ограничения, квантовый отжиг остаётся перспективным направлением для экспериментов. Существующие машины для квантового отжига, такие как устройства компании D-Wave (см., например, [19]), уже подходят для исследований и могут быть использованы для изучения возможностей реализации машин с передачей данных из будущего.

5.4. Итог

Реализация машины Тьюринга с каналом передачи данных во времени возможна с использованием различных подходов, включая детерминированный поиск с возвратом, методы отжига и, в частности, квантовый отжиг. Каждый из этих методов имеет свои преимущества и ограничения, и выбор конкретного подхода зависит от характеристик решаемой задачи и доступных вычислительных ресурсов. Квантовый отжиг, в частности, остаётся перспективным направлением, хотя его практическая эффективность требует дальнейшего изучения.

6. Конструктивные распознаватели: пример решения задачи на машине с передачей данных во времени

6.1. Неконструктивность классических распознавателей

Классические алгоритмы распознавания обычно неконструктивны: они выдают только бинарный ответ (ДА или НЕТ) без объяснения результата. Например, на вопрос от том, существует ли y, удовлетворяющее условию $P(x, y)$, такие алгоритмы лишь подтверждают или опровергают существование такого y, не предоставляя сам объект y при положительном ответе или доказательство его отсутствия при отрицательном. Это ограничивает их использование в задачах, где требуется обоснование решения.

6.2. Конструктивные задачи в классе NP ∩ co-NP

Машина Тьюринга с каналом передачи данных во времени позволяет реализовать конструктивные распознаватели, выдающие не только ответ, но и сертификат — доказательство или свидетельство результата. Мы рассматриваем симметричные задачи класса $\mathsf{NP} \cap \mathsf{co\text{-}NP}$, где свойство $Q(x)$ истинно, если существует сертификат y такой, что $Q_1(x, y)$, и ложно, если существует сертификат z такой, что $Q_0(x, z)$, причём размеры y и z полиномиально ограничены относительно $|x|$.

6.3. Пример: проверка наличия нетривиального делителя

Рассмотрим задачу определения, имеет ли число x нетривиальный делитель y такой, что $1 < y \leq d$ (порог d задан). Это свойство $Q(d, x)$:

- Если $Q(d, x) = 1$, то сертификат — делитель y, проверяемый делением за полиномиальное время.
- Если $Q(d, x) = 0$, то сертификат — факторизация x на простые множители, все $> d$, проверяемая за полиномиальное время (например, с использованием теста Миллера [20] при допущении обобщённой гипотезы Римана).

6.4. Реализация на машине с передачей данных во времени

Конструктивный распознаватель на машине с передачей данных во времени работает следующим образом:

(1) *Получение данных из будущего*:

- На первом шаге машина запрашивает с меткой 0 пару (b, C), где $b \in \{0, 1\}$ — ответ, а C — сертификат (делитель или факторизация).
- Ответ b немедленно записывается на выходную ленту (если не требуется выдача ответа с фиксированной задержкой, то можно выдать и сертификат, тогда распознаватель получится конструктивным).

(2) *Проверка сертификата*:

- Если $b = 1$, то проверяется, что $C = y$ делит x и $1 < y \leq d$.
- Если $b = 0$, то проверяется, что C — факторизация x на простые множители $> d$, и их произведение равно x.

(3) *Самосогласование*:

- После проверки, если сертификат корректен, то (b, C) записывается на коммуникационную ленту с меткой 0.
- При некорректности создаётся противоречие, и ветвь вычислений отбрасывается.

Нетрудно видеть, что эта машина работает полиномиальное время.

Примеры выполнения:

Вход: $x = 15$, $d = 4$.

Шаг 1: Импорт $(1, 3)$ из будущего, вывод 1.
Шаг 2: Проверка: $15 \div 3 = 5$, $1 < 3 \leq 4$, корректно.
Шаг 3: Передача $(1, 3)$ с меткой 0, самосогласование выполнено.

Вход: $x = 17$, $d = 4$.

Шаг 1: Импорт $(0, \{17\})$, вывод 0.
Шаг 2: Проверка: 17 — простое (тест Миллера), $17 > 4$, корректно.
Шаг 3: Передача $(0, \{17\})$, самосогласование выполнено.

6.5. Особенности и преимущества

Линейность: Процесс детерминирован.

Мгновенный отклик: Ответ доступен на первых шагах благодаря передаче из будущего.

Прозрачность: Если выдаётся сертификат, то это делает результат интерпретируемым, усиливая наглядность вычислений.

На рисунке 3 представлена схема работы конструктивного распознавателя: данные (b, C) передаются из конца вычислений в начало для немедленного вывода и последующей проверки. Блок-схема процесса от импорта результата до проверки и самосогласованной передачи данных дана на рисунке 4.

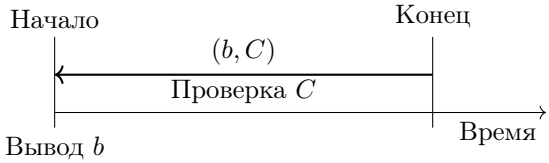

Рисунок 3. Схема работы конструктивного распознавателя

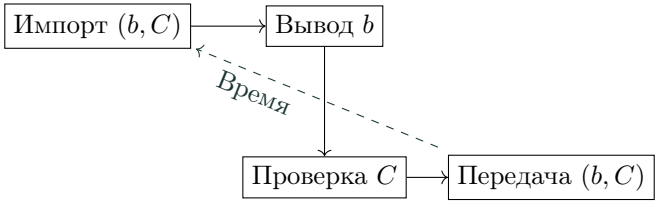

Рисунок 4. Блок-схема процесса

7. Действия для передачи данных во времени

Для преобразования обычного языка программирования в язык, поддерживающий передачу данных во времени, необходимо добавить две новые операции: передача данных в прошлое и приём данных из будущего. Эти действия позволяют машине взаимодействовать как с будущими, так и с прошлыми состояниями вычислительного процесса, предоставляя новые возможности для программирования. В данном разделе мы рассмотрим эти действия и их роль в организации вычислений с передачей данных во времени.

7.1. Действие передачи данных в прошлое

Действие передачи данных в прошлое позволяет машине записать данные в определённый момент времени, которые могут быть использованы в прошлых вычислениях. Это действие формализуется следующим образом:

Формальное описание:

- *transfer*(l, d): передаёт объект d в момент времени l.
  - l — метка момента времени, для который передаются данные.
  - d — передаваемый объект (например, бит, число или строка).

Принцип работы:

(1) Машина записывает объект d в канал передачи данных во времени с меткой времени l.

(2) Данные становятся доступными для использования в прошлых шагах вычислений, связанных с моментом времени l.

(3) Если запись с меткой l уже существует, возникает ошибка.

Пример: Если машина выполняет действие *transfer*$(1, 1)$, значение 1 передаётся с меткой 1. В прошлых вычислениях, связанных с 1, это значение может быть использовано для принятия решений.

7.2. Действие приёма данных из будущего

Действие приёма данных из будущего позволяет машине получить данные из определённого момента в будущем, что делает возможным использование будущей информации в текущих вычислениях. Это действие формализуется следующим образом:

Формальное описание:

- *accept*(l): принимает данные из момента времени с меткой l.
  - l — имя момента времени, из которого запрашиваются данные.

Принцип работы:

(1) Машина запрашивает данные из канала передачи данных во времени с меткой времени l.

(2) Если данные существуют, они возвращаются и могут быть использованы в текущих вычислениях.

(3) Если данные отсутствуют, возникает ошибка.

Пример: Если машина выполняет действие *accept*(1), она получает значение d, которое было передано с меткой 1 с помощью действия *transfer*$(1, d)$. Это значение затем может быть использовано для принятия решений в текущих вычислениях.

7.3. Взаимодействие передач данных

Действия передачи и приёма данных работают совместно, обеспечивая связь между текущими и будущими состояниями вычислительного процесса. Это взаимодействие позволяет машине получать данные из будущего для принятия решений в текущих вычислениях. Важно отметить, что объём данных, передаваемых во времени, является важным показателем сложности вычисления. Например, при реализации процесса поиском с возвратом время работы замедляется экспоненциально с двоичным показателем объёма данных в битах, так как для каждого бита данных фактически приходится ввести две ветви вычислений. Подобные сложности возникают и в методах отжига.

7.4. Итог

Действия передачи и приёма данных во времени являются ключевыми элементами нашей вычислительной модели. В принципе, мы можем включить эти возможности в любой язык программирования, превратив его в язык управления машиной с передачей данных из будущего в прошлое. Эти инструменты предоставляют новые возможности для создания интуитивных вычислительных систем.

8. Распараллеливание вычислительного процесса

В данном разделе описывается параллельная машина с передачей данных во времени — расширение нашей машины с дополнительной возможностью, направленной на увеличение её вычислительной мощности: выполнение другого экземпляра машины с определёнными входными данными. Строка входных данных для нового процесса помещается на специальную ленту аргументов. Параллельный процесс запускается при переходе машины в специальное состояние запуска параллельного процесса. Новый процесс начинается с особого начального состояния подпроцесса. Изначально новый процесс выполняется до тех пор, пока не достигнет состояния, в котором выводит результат (в то время как старый процесс ожидает). Этот результат затем передаётся с выходной ленты нового процесса на ленту аргументов старого процесса. Затем оба процесса продолжают выполняться параллельно, при этом старый процесс больше не взаимодействует с новым процессом. Параллельное выполнение нового процесса необходимо для передачи данных в прошлое внутри него, что обеспечивает корректность результата, производимого этим процессом.

Новый процесс может быть запущен столько раз, сколько необходимо, основным процессом, либо порождёнными подпроцессами. Поскольку подпроцессы могут запускать свои собственные подпроцессы параллельно и независимо, количество подпроцессов может расти экспоненциально по мере работы машины в параллельном режиме.

Общее параллельное время работы машины считается временем до завершения всех параллельных процессов, как если бы они выполнялись синхронно, шаг за шагом.

Теорема 3. *На параллельной машине с передачей данных во времени можно за полиномиальное время промоделировать работу обычной машины Тьюринга с полиномиальной памятью (что соответствует классу сложности* PSPACE*).*

Доказательство. Рассмотрим машину Тьюринга с известной оценкой её емкостной сложности в классе PSPACE. Выполним следующее.

(*1*) Сначала оценим время работы этой машины (экспоненциальное от объёма памяти).

(*2*) Запустим подпроцесс для моделирования работы машины в течение половины этого времени (время моделирования передаётся подпроцессу вместе с начальной конфигурацией процесса).

(*3*) Получив результат и позволив подпроцессу продолжить для его формирования в будущем, немедленно запустим новый подпроцесс для моделирования работы исходной машины в течение оставшейся второй половины выделенного времени, передав этому подпроцессу конфигурацию, полученную от первого подпроцесса.

(*4*) Выведем результат.

(*5*) Подпроцессы продолжают работу аналогичным образом, пока выделенное время не сойдётся к заранее определённой константе.

Нетрудно видеть, что глубина вызовов подпроцессов здесь полиномиальна, и общее параллельное время работы также полиномиально. □

Теорема 4. *Работа параллельной машины (с передачей данных во времени) за полиномиальное время может быть промоделирована обычной машиной Тьюринга с полиномиальным объёмом памяти.*

Доказательство. Обратное утверждение следует из того, что работу такой параллельной машины можно промоделировать на обычной машине Тьюринга с использованием полиномиального объёма памяти. Это

достигается за счёт последовательного выполнения всех запущенных процессов, так как они не взаимодействуют друг с другом. Поскольку общее параллельное время моделируемой машины ограничено полиномиально, глубина вызовов подпроцессов также ограничена полиномиально. Таким образом, общий объём памяти для моделирования также остаётся полиномиальным. □

9. Прямой доступ к входным данным

В данном разделе мы модифицируем модель нашей машины, добавив возможность работы с большими данными через прямой доступ к ячейкам входной ленты. Под большими данными понимаются данные, которые невозможно последовательно просмотреть в рамках текущих временных ограничений. Механизм прямого доступа позволяет быстро обращаться к удалённым ячейкам, что позволяет преодолеть указанное ограничение. Это достигается следующим образом: вводится специальная адресная лента, на которой записывается двоичное представление номера ячейки на входной ленте. Головка входной ленты мгновенно перемещается к указанной ячейке. Другие механизмы перемещения этой головки не предусмотрены. Мы предполагаем, что все запущенные подпроцессы работают с одними и теми же входными данными. Таким образом, мы определили параллельную машину с передачей данных во времени и с прямым доступом ко входным данным.

Если заменить PSPACE на LOGSPACE в предыдущем разделе, мы получим следующую теорему.

Теорема 5. *На параллельной машине с передачей данных во времени и с прямым доступом к входным данным работа обычной машины Тьюринга с логарифмическим объёмом памяти (*LOGSPACE*) может быть промоделирована за логарифмическое время (*LOGTIME*).*

Доказательство. Рассуждение аналогично доказательству Теоремы 3, но с заменой полиномиальных ограничений на логарифмические. Поскольку доступ к входным данным происходит за постоянное время, общее время выполнения остаётся логарифмическим. □

Обратное утверждение проблематично, поскольку прямое моделирование логарифмического времени на нашей машине с использованием обычной машины Тьюринга требует объёма памяти, полиномиального от логарифма размера исходных данных. Однако аналогичный подход может быть использован для доказательства следующей теоремы.

Теорема 6. *Класс* POLYLOGTIME *на параллельной машине с передачей данных во времени и с прямым доступом к входным данным совпадает с классом* POLYLOGSPACE *для обычных машин Тьюринга.*

Доказательство. Утверждение основано на том, что работа нашей машины с прямым доступом может быть промоделирована на обычной машине Тьюринга с использованием полилогарифмической памяти. Таким образом, общее время моделирования остаётся полилогарифмическим. □

Заключение

В данной работе предложена принципиально новая вычислительная модель, расширяющая классические представления о вычислительных процессах за счёт механизма передачи данных состояниями машины в различные моменты времени. Основные результаты исследования заключаются в следующем:

(1) *Классы сложности и временные ограничения*: Доказано, что машина с передачей данных из будущего, без ограничений по времени, распознаёт в точности разрешимые множества с постоянной задержкой ответа. При полиномиальных ограничениях по времени модель соответствует классу задач из пересечения NP и co-NP. Эти результаты показывают, как контролируемые нарушения причинного порядка могут помогать вычислениям.

(2) *Распараллеливание и емкостная сложность*: Расширение модели возможностью запуска подпроцессов, наследующих входные данные, позволяет охватить класс PSPACE за полиномиальное время, что демонстрирует, как временная сложность преобразуется в емкостную.

(3) *Прямой доступ и эффективность обработки данных*: Модификация с адресуемым доступом ко входной ленте и логарифмической временной сложностью обеспечивает соответствие классу LOGSPACE, что особенно актуально для задач обработки больших данных. Этот результат поддерживается механизмом мгновенного перемещения головки входной ленты, устраняющим линейные задержки.

(4) *Практическая реализация*: Предложены два подхода к практической реализации модели:

- *Детерминированный поиск с возвратом* — систематический поиск самосогласованных вычислительных траекторий;

- *Методы отжига* (включая квантовые системы) — поиск состояний с минимальной энергией противоречий.

Эксперименты с квантовыми отжигателями, такими как D-Wave, могут стать следующим шагом для оценки физической реализуемости модели.

Перспективы дальнейших исследований охватывают как теоретические, так и практические аспекты:

- Оптимизация объёма передаваемых данных для снижения вычислительной нагрузки;
- Разработка специализированных языков программирования с операторами для передачи данных во времени;
- Анализ связи с квантовыми вычислениями.

Данная работа актуальна не только для теоретической информатики, но и для междисциплинарных исследований. Она демонстрирует, что связь состояний во времени может стать ключом к решению задач, недоступных для традиционных систем. Предложенная модель может служить мостом между теорией сложности и физическими реализациями, открывая новые возможности для алгоритмов.

Список использованных источников

[1] Cook S. A. *The complexity of theorem-proving procedures* // *Logic, automata, and computational complexity: The works of Stephen A. Cook*, ed. Kapron B.C., New York: ACM.– 2023.– ISBN 979-8-4007-0779-7.– Pp. 143–152. ↑31

[2] Karp R. M. *Reducibility among combinatorial problems* // *50 Years of Integer Programming 1958–2008. From the Early Years to the State-of-the-Art*, eds. Jünger M., et al., Berlin–Heidelberg: Springer.– 2010.– ISBN 978-3-540-68274-5.– Pp. 219–241. ↑31

[3] Garey M. R., Johnson D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Series of Books in the Mathematical Sciences.– San Francisco: Freeman.– 1979.– ISBN 978-0716710455.– 340 pp. ↑32

[4] Aaronson S. *Guest column: NP-complete problems and physical reality* // ACM Sigact News.– 2005.– Vol. **36**.– No. 1.– Pp. 30–52. ↑32

[5] Deutsch D. *Quantum theory, the Church–Turing principle and the universal quantum computer* // Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences.– 1985.– Vol. **400**.– No. 1818.– Pp. 97–117. ↑32

[6] Brun T. A. *Computers with closed timelike curves can solve hard problems efficiently* // Foundations of Physics Letters.– 2003.– Vol. **16**.– No. 3.– Pp. 245–253. doi ↑32

[7] Aaronson S., Watrous J. *Closed timelike curves make quantum and classical computing equivalent* // Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences.– 2009.– Vol. **465**.– No. 2102.– Pp. 631–647. doi ↑32

[8] Papadimitriou Ch. *Computational Complexity.*– Reading, Massachusetts: Addison-Wesley.– 1994.– ISBN 978-0201530827.– 523 pp. ↑32

[9] Arora S., Barak B. *Computational Complexity: A Modern Approach.*– New York: Cambridge University Press.– 2009.– ISBN 978-0521424264.– 594 pp. ↑32

[10] Nielsen M. A., Chuang I. L. *Quantum computation and quantum information.*– New York: Cambridge University Press.– 2010.– ISBN 9781107002173.– 702 pp. doi ↑32

[11] Grover L. K. *A fast quantum mechanical algorithm for database search* // *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (22–24 May 1996, Pennsylvania, USA), New York: ACM.– 1996.– ISBN 978-0-89791-785-8.– Pp. 212–219. doi ↑32

[12] Shor P. W. *Algorithms for quantum computation: discrete logarithms and factoring* // *Proceedings 35th Annual Symposium on Foundations of Computer Science* (20–22 November 1994, Santa Fe, NM, USA).– IEEE.– 1994.– ISBN 0-8186-6580-7.– Pp. 124–134. doi ↑32

[13] Harrow A. W., Montanaro A. *Quantum computational supremacy* // Nature.– 2017.– Vol. **549**.– No. 7671.– Pp. 203–209. doi ↑32

[14] Iqbal K., Ahmad O., Vandika A. Y. *Quantum computing and its implications for complex system analysis* // Research of Scientia Naturalis.– 2024.– Vol. **1**.– No. 5.– Pp. 238–247. doi ↑32

[15] Orlicki J. I. *Time-travelling Turing Machines and the self-consistent Halting Problem.*– 2024. doi ↑32

[16] Gödel K. *An example of a new type of cosmological solutions of Einstein's field equations of gravitation* // Rev. Mod. Phys.– 1949.– Vol. **21**.– No. 3.– Pp. 447–450. doi ↑33

[17] Bacon D. *Quantum computational complexity in the presence of closed timelike curves* // Phys. Rev. A.– 2004.– Vol. **70**.– No. 3.– id. 032309. doi ↑33

[18] Wolpert D. H. *Physical limits of inference* // Physica D: Nonlinear Phenomena.– 2008.– Vol. **237**.– No. 9.– Pp. 1257–1281. doi ↑33

[19] King A. D., Raymond J., Lanting T., Harris R., Zucca A., Altomare F., Berkley A. J., Boothby K., Ejtemaee S., Enderud C., Hoskinson E., Huang Sh., Ladizinsky E., MacDonald A. J. R., Marsden G., Molavi R., Oh T., Poulin-Lamarre G., Reis M., Rich Ch., Sato Y., Tsai N., Volkmann M., Whittaker J. D., Yao J., Sandvik A. W., Amin M. H. *Quantum critical dynamics in a 5,000-qubit programmable spin glass* // Nature.– 2023.– Vol. **617**.– No. 7959.– Pp. 61–66. doi ↑33, 43

[20] Miller G. L. *Riemann's hypothesis and tests for primality* // *STOC '75: Proceedings of the seventh annual ACM symposium on Theory of computing* (5–7 May 1975, Albuquerque, New Mexico, USA), New York: ACM.– ISBN 978-1-4503-7419-4.– С. 234–239. ↑44

Поступила в редакцию 15.03.2025;
одобрена после рецензирования 02.04.2025;
принята к публикации 02.04.2025;
опубликована онлайн 21.04.2025.

Рекомендовал к публикации *д.ф.-м.н. Н. Н. Непейвода*

Информация об авторах:

Милад Джудакизаде

Аспирант в области искусственного интеллекта и машинного обучения, исследования сосредоточены на вычислительной сложности, машинном обучении и логике в компьютерных науках.

0000-0002-6167-6237
e-mail: joudakizadeh@mail.ru

Анатолий Петрович Бельтюков

Доктор физико-математических наук, профессор. Научные интересы включают вычислительную сложность, классы сложности, субрекурсивные иерархии, интуиционистскую математику, конструктивные системы, механизацию доказательств, сложность доказательств, модели вычислений, субструктурные логики, слабые арифметики и логику в компьютерных науках.

0000-0002-3433-9067
e-mail: **belt.udsu@mail.ru**

Авторы внесли равный вклад в подготовку публикации.

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*