Proof Sketch. The idea is to construct a "universal" algorithm that simulates all possible algorithms in parallel, allocating more and more time to each algorithm as the computation progresses.

Let $\{M_i\}$ be an enumeration of all algorithms (e.g., all Turing machines). We construct an algorithm U that works as follows:

For $t = 1, 2, 3, ...$: For $i = 1, 2, ..., t$: Run M_i on input x for $2^{i-t}t$ steps If M_i outputs a y such that $A(x, y)$ is true, return y

If there exists an algorithm that solves $A(x, y)$ in time $T(n)$, then U will find a solution in time $O(T(n))$. The multiplicative constant comes from the overhead of simulating multiple machines, and the additive term comparable to the length of x comes from the initial steps where t is small.

This algorithm is optimal up to a constant factor because if there were a significantly faster algorithm, it would contradict the assumption that $T(n)$ was the time of the fastest algorithm. □

Acknowledgements

The author expresses sincere gratitude to A. N. Kolmogorov, B. A. Trakhtenbrot, Ya. M. Barzdin, Yu. I. Albryton, and M. I. Degtyar for valuable discussion.

References

[1] Yablonsky S. V. On algorithmic difficulties of synthesis of minimal contact circuits. Collection "Problems of Cybernetics", 2. Moscow, Fizmatgiz, 1959, 75-121.

[2] Zhuravlev Yu. I. Set-theoretic methods in Boolean algebra. Collection "Problems of Cybernetics", 8. Moscow, Fizmatgiz, 1962, 5-44.

[3] Trakhtenbrot B. A. Optimal computations and Yablonsky's frequency phenomenon. Seminar. Novosibirsk, "Nauka", Siberian Branch, 1965, 4, 5, 79-93.

[4] Degtyar M. I. On the impossibility of eliminating complete enumeration when calculating functions relative to their graphs. Reports of the USSR Academy of Sciences, 1969, 189, 4, 748-751.