On The Limitations of Self-Sustaining Machines

Robert Mapes

September 25, 2024

Abstract

This paper argues that no machine system can be fully self-sustaining, as unfixable states inevitably arise when machines attempt to repair each other. Drawing on Gödel's incompleteness theorem and Turing's Halting Problem, we assert that machines are constrained by their design and require external intervention to overcome these limitations.

Contents

1 Introduction

The idea that machines could become fully autonomous and self-sustaining has been a long-standing goal in the fields of artificial intelligence (AI) and robotics. However, we argue that due to fundamental limitations in both logic and the process of machine creation, no machine can ever completely replace the need for human intervention. The human, as the creator, inherently limits the machine's capabilities to a defined set of operations, making certain failure states unavoidable.

This idea extends from both theoretical frameworks like Gödel's incompleteness theorem and practical considerations of machine fault tolerance. Machines are created based on finite rules, algorithms, and structures defined by humans. We demonstrate that the human operator H, with capabilities outside the machine's formal system, is essential for resolving certain failures that no machine within the system can fix.

2 Formalizing the System of Machines TM^*

Consider a system of machines $\{M_0, M_1, \ldots, M_n\}$, where each machine M_i is modeled as a variant of a Turing machine, tasked with repairing another machine M_k in case of failure. Each machine is described by the tuple:

$$TM_i^* = (Q_i, \Sigma_i, \Gamma_i, \delta_i, q_{0,i}, q_{r,i}, F_i)$$

Where:

- Q_i: Finite set of states for machine M_i.
- Σ_i: Input alphabet.
- $\Gamma_i \supseteq \Sigma_i$: Tape alphabet, including a blank symbol $\#$.
- $\delta_i : Q_i \times \Gamma_i \to Q_i \times \Gamma_i \times \{L, R\}$: Transition function.
- $q_{0,i} \in Q_i$: Initial state.
- $q_{r,i} \in Q_i$: Repair state, where M_i attempts to fix a failure.
- $F_i \subseteq Q_i$: Set of halting and failure states.

(Placeholder): Provide a detailed explanation of the significance of each component in the tuple TM_i^* and how they interact. This could include examples of different configurations.

2.1 Working and Non-working States

We define two essential sets of tape positions for each machine:

- P_W: Set of tape positions corresponding to working states, indicating that machine M_i is functional.
- P_{NW}: Set of tape positions corresponding to non-working states, indicating that machine M_i is non-functional.

Formally, these positions can be defined as subsets of the tape:

$$P_W \subseteq \mathrm{T}$$
$$P_{NW} \subseteq \mathrm{T}$$

These two sets are disjoint:

$$P_W \cap P_{NW} = \emptyset$$

and together they partition the set of tape positions:

$$P_W \cup P_{NW} = \mathrm{T}$$

When a machine reads a symbol from the tape, it can transition into a non-working position P_{NW} based on the symbol and its current state. This behavior is formalized as:

$$\text{If } M_i \text{ reads a symbol } s \in \Gamma_i \text{ at position } p, \text{ it will produce a non-working result if:}$$

$$P_{NW} = \{p \in \text{Tape} \mid \delta_i(q, s) = (q', s', R) \text{ such that } q' \in NW\}$$

This means that when the machine encounters a specific symbol s at position p, it may transition into a non-working state P_{NW} without halting its operation. Consequently, the machine's behavior is characterized by both its operational state and the current configuration of the tape.

(Placeholder): Discuss potential overlaps or gray areas between P_W and P_{NW}, addressing edge cases where a machine might be in transition between states.

2.2 The Repair Function $F(TM_i^*)$

The repair function governs how a non-working machine can be restored to a working state. Specifically, it defines how a machine M_k attempts to repair machine M_i by transitioning from non-working tape positions (P_{NW}) to working tape positions (P_W) through modification of the tape symbols.

The repair function is rigorously defined as:

$$F : P_{NW} \times \Gamma_i \to P_W \times \Gamma_i$$

This mapping specifies how machine M_k intervenes in the operation of machine M_i to correct any failure by modifying the tape at the location where M_i encountered a failure, restoring M_i to a working state. The repair function is defined as follows:

$$F(TM_i^*) = \begin{cases} (p_b, s') & \text{if } M_i \in P_{NW} \text{ and conditions } C_{repair} \text{ are met} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Where:

- $p_b \in P_W$: A tape position in the working state P_W to which the machine transitions after repair.
- $s' \in \Gamma_i$: The new symbol on the tape that replaces the original symbol at the failure point, allowing the machine to resume operation.

(Example Placeholder): An example could be provided here where machine M_i fails at position $p_a \in P_{NW}$ with input symbol s, and machine M_k successfully replaces s with a corrected symbol s', restoring the machine to $p_b \in P_W$.

The success of the repair function relies on meeting a set of conditions, denoted as C_{repair}, which ensures that the transition from a non-working to a working state is feasible and valid.

2.2.1 Conditions for Repair Success

The conditions for successful repair are defined as:

$$C_{repair} = \{c_1, c_2, c_3, \dots, c_n\}$$

The following conditions are critical for the repair function to succeed:

- c_1: The current tape position p_a corresponding to machine M_i must be identifiable within the set of non-working positions P_{NW}. In other words, $p_a \in P_{NW}$, meaning machine M_i is indeed in a failure state.
- c_2: The symbol $s \in \Gamma_i$ at the current tape position must allow for a valid transition back to a working state. Formally, $\exists(q_a, s) \in \delta_i$ such that there exists a transition to (p_b, s', R) with $p_b \in P_W$, meaning there is a defined path from the failure state back to a working state.
- c_3: The machine M_i must have sufficient computational resources available to perform the repair. This can be expressed as:

  $$\text{resources}(M_i) \geq R_{min}$$

  where R_{min} is the minimum required resources (e.g., time steps, tape length, memory).

- c_4: The repair must not introduce an infinite loop. This ensures that the machine doesn't enter a state where it continuously attempts repair without resolving the failure.
- c_5: The symbol replacement must be valid within the context of the machine's transition function δ_i. That is, $s' \in \Gamma_i$ must lead to a valid transition that returns the machine to a working state.
- c_6: Any dependencies on prior tape positions or outputs must be resolved. This ensures that the repair does not trigger new failures or inconsistencies.

Additional conditions could be included here depending on the complexity of the machine's operation or environmental constraints (e.g., time limitations, external inputs).

2.2.2 Failure Cases and Limitations

(Placeholder): A detailed discussion of failure cases should be included here. For example, if condition c_2 is not met and no valid transition exists from the failure position to a working position, the repair function will be undefined, and machine M_k will be unable to restore M_i to a working state.

2.2.3 Example of Repair Function in Action

To illustrate how the repair function operates, consider a scenario where machine M_i fails at position $p_a \in P_{NW}$ due to an incorrect tape symbol s. Machine M_k inspects the tape and determines that replacing s with s' will allow M_i to transition to position $p_b \in P_W$. The transition function of machine M_i specifies:

$$\delta_i(p_a, s) = (p_b, s', R) \quad \text{where } p_b \in P_W$$

In this case, the repair function $F(TM_i^*)$ succeeds, as the new symbol s' leads to a valid working state.

If any of the conditions C_{repair} were not satisfied—such as insufficient resources (c_3) or the absence of a valid transition (c_2)—the repair attempt would fail.

2.2.4 Final Considerations

This repair function ensures that non-working positions are systematically addressed, with specific conditions guiding when a machine can be successfully repaired. Future iterations of the repair function could consider additional complexities, such as interactions with other machines or time constraints on repairs.

2.3 Definition Of The Unfixable State

An unfixable state occurs when a machine reaches a failure condition that no subsequent machine can repair. For example, if a machine M_n enters the non-working positions P_{NW}, it requires intervention from another machine M_k. However, if M_k is unable to effect a repair, it may itself transition into a non-working position, thereby becoming a member of the set of non-working positions P_{NW_k}. Consequently, for machine M_n, no other machine M_k can transition M_n from the non-working positions P_{NW} back to the working positions P_W. Formally, this is defined as:

$$F(TM_n^*) = \emptyset \quad \text{(undefined repair function)}$$

This implies that there exists no valid mapping for the repair function F that allows M_k to restore M_n to a functional state.

- The internal state of M_n has reached a configuration that is not addressable by M_k.
- Multiple machines are in non-working positions simultaneously, creating a cascading failure that exceeds the repair capabilities of subsequent machines.
- The necessary resources to effect a repair are exhausted or cannot be met by the subsequent machines.
- **(Placeholder)**: Discuss exceptional cases, such as failure modes where repairs are impossible despite available resources.
- **(Placeholder)**: Analyze how changes in system parameters (e.g., resource availability, state configurations) can affect the ability to repair.

2.4 Preliminary Lemmas

To establish the unsustainability of the system, we propose the following preliminary lemma:

Lemma 2.1 *If a machine M_i reaches an unfixable state P_{NW_i} such that $F(TM_i^*) = \emptyset$, then all subsequent machines $M_{i+1}, M_{i+2}, \ldots$ cannot transition M_i back to a working state.*

(Placeholder): Add a lemma regarding the conditions that lead to an unfixable state, such as specific configurations of P_W and P_{NW}.

(Placeholder): Add a lemma that examines the impact of resource constraints on the ability of M_{i+1} to repair M_i, establishing a relationship between resource availability and repair success.

(Placeholder): Include a lemma that discusses the interactions between multiple machines.

3 Gödel's Incompleteness Theorem and Turing Machines TM^*

In this section, we will formally prove that Gödel's incompleteness theorem holds for the Turing machine variant TM^*. We will explore the implications of this theorem for Turing machines, demonstrate a modeled example of a TM^* encountering incompleteness, and introduce the Halting Problem as a significant instance that all Turing machines must contend with.

3.1 Formal Proof of Gödel's Incompleteness for TM^*

Let TM^* be a Turing machine defined by a set of states Q, a tape alphabet Γ, a transition function $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, and an initial state $q_0 \in Q$. We assume TM^* is capable of simulating any computation that a standard Turing machine can perform.

Gödel's first incompleteness theorem states that in any consistent formal system S that is capable of expressing basic arithmetic, there exist true statements that cannot be proven within that system. For TM^*, we can formalize this as follows:

1. **Assumption of Consistency**: Assume that the formal system S_H that defines the operations of TM^* is consistent. 2. **Existence of True but Unprovable Statements**: According to Gödel's theorem, there exists a statement P such that:

$$P \in \text{True}(S_H) \wedge P \notin \text{Provable}(S_H)$$

This implies that TM^* cannot construct a proof for statement P within its operational framework. 3. **Implication for TM^***: Thus, for some computation C performed by TM^*, we find that:

$$\exists C \quad \text{such that} \quad C \text{ is true but not provable in } S_H$$

Therefore, we conclude that Gödel's incompleteness theorem holds for the Turing machine variant TM^*.

3.2 Implications of Gödel's Incompleteness for Turing Machines

The implications of Gödel's theorem for Turing machines are profound:

1. **Inherent Limitations**: Just as formal systems can contain true statements that are unprovable, Turing machines are similarly constrained. They can encounter states or problems that are beyond their capacity to resolve. 2. **Need for External Intervention**: When a TM^* encounters such a state, it necessitates human intervention or a mechanism beyond its own logic to resolve the incompleteness, reflecting the need for a "fix function."

3.3 Modeled Example of Incompleteness in TM^*

Consider a TM^* tasked with evaluating whether a given statement S is true within the framework S_H. Suppose S encodes a self-referential statement, similar to Gödel's construction:

$$S \text{ states, "This statement is unprovable."}$$

In this scenario: - The machine processes the statement but finds that:

$$S \in \text{True}(S_H) \wedge S \notin \text{Provable}(S_H)$$

- Consequently, the TM^* reaches an unfixable state S_M where it cannot determine the truth value of S, leading to a failure to provide a conclusive output.

In this situation, the human operator H must intervene, employing the fix function F_H to guide the machine out of its unfixable state S_M:

$$F_H(TM^*) : S_M \to W$$

3.4 The Halting Problem as a Key Example

The Halting Problem serves as a paramount example of the limitations of Turing machines, illustrating Gödel's incompleteness theorem in a practical context. The Halting Problem posits that there is no algorithm that can universally determine whether a Turing machine will halt on a given input. Formally, we can state:

$$\neg\exists H : \{(M, x) | M \text{ halts on input } x\} \text{ is computable}$$

For any TM^* and input x, there exist configurations that lead the machine to an infinite loop or an ambiguous operational state, indicating that: - There are cases where TM^* will enter an unresolvable state, similar to the unprovable statements derived from Gödel's theorem. - The need for human intervention becomes evident when TM^* faces these limitations, confirming that no machine can achieve full autonomy or self-sustainability.

Thus, the exploration of Gödel's incompleteness theorem highlights the fundamental limitations of Turing machines and reinforces the necessity for external intervention to navigate complexities that arise from their inherent constraints.

4 Halting Problem for TM^* with Unfixable and Fix States

Let TM^* be a variant of a Turing machine with two key additional properties:

- The machine can encounter an **unfixable state** where further computation cannot continue in a meaningful or defined way. - A **fix function** may attempt to correct the machine's state. - The machine may enter a **fix state** if the fix function is applied, but may still not halt if this fix does not resolve the issue. - The machine may enter a **nonworking/halting state** indicating that computation either halts properly or enters a state of failure.

We analyze the Halting Problem for this extended TM^* and the behavior of these additional states in the context of halting.

4.1 Definitions and Problem Formulation

Given a Turing machine TM and input w, we denote the problem as determining whether $TM(w)$ halts. Formally, the function $H(TM, w)$ is defined as:

$$H(TM, w) = \begin{cases} 1 & \text{if } TM(w) \text{ halts,} \\ 0 & \text{if } TM(w) \text{ runs forever.} \end{cases}$$

For the variant TM^*, the machine is equipped with a **fix function** Fix, which is invoked when the machine enters an unfixable state. The system can be described as a combination of standard Turing machine configurations, along with states that reflect potential failure or repair attempts.

4.2 Unfixable State and Fix Function

Define the **unfixable state** as a special state $q_u \in Q$ such that no valid transitions can occur from this state in the usual computation. Formally, we define the unfixable state as:

$$q_u \in Q \text{ where } \delta(q_u, s) = \emptyset \quad \forall s \in \Gamma.$$

The **fix function** $\text{Fix}(q_u, t)$ is defined as an attempt to transition the machine from an unfixable state q_u to a new state $q_f \in Q$ (called the **fix state**) based on certain conditions $t \in T$, where T is a set of fixable conditions. Thus:

$$\text{Fix}(q_u, t) = q_f, \quad \text{if } t \text{ is satisfied.}$$

The machine can thus move from the unfixable state to the fix state q_f, where further computation may resume if t is true.

4.3 Simulation with Unfixable and Fix States

Now consider the machine TM^* simulating TM on input w, represented by the configuration sequence:

$$C_0 \xrightarrow{\delta} C_1 \xrightarrow{\delta} C_2 \xrightarrow{\delta} \cdots \xrightarrow{\delta} C_n.$$

If at some step i, the machine enters an unfixable state q_u, the fix function $\text{Fix}(q_u, t)$ is invoked:

$$C_i = (q_u, \gamma_i) \quad \Rightarrow \quad C_i' = (q_f, \gamma_i') \quad \text{if Fix}(q_u, t) \text{ applies.}$$

Here, γ_i and γ_i' represent the respective tape contents before and after the fix function is applied. If t is not satisfied, the fix fails, and the machine remains in q_u indefinitely.

The fix state q_f may allow for the continuation of computation, or it may lead to a **nonworking state** q_n, defined as:

$$\delta(q_f, s) = q_n \quad \text{for some } s \in \Gamma.$$

If the machine enters q_n, it signals that further computation cannot proceed meaningfully, akin to a halting condition.

4.4 Halting and Non-Halting in TM^*

If TM^* reaches a halting state q_h, then:

$$TM^*(w) = H(TM, w) = 1.$$

However, the introduction of the unfixable state and fix state complicates the determination of halting. Specifically, we now face the following possibilities:

1. **Halting**: TM^* reaches q_h after a finite number of steps. 2. **Non-Halting**: TM^* enters an unfixable state q_u, and the fix function fails, leaving the machine in q_u or transitioning it to q_n. 3. **Fixable Computation**: TM^* enters an unfixable state q_u, but the fix function successfully transitions the machine to q_f, allowing further computation and possibly halting later.

Formally, we describe the behavior of TM^* using the function $F(TM^*, w)$, which captures the interaction between halting, unfixable, and fixable states:

$$F(TM^*, w) = \begin{cases} 1 & \text{if } TM^*(w) \text{ halts,} \\ -1 & \text{if } TM^*(w) \text{ enters } q_n, \\ \infty & \text{if } TM^*(w) \text{ remains in } q_u. \end{cases}$$

4.5 Undecidability with Fix and Unfixable States

We now examine whether the introduction of unfixable and fix states changes the decidability of the Halting Problem. Even with the fix function, TM^* cannot always decide whether $TM(w)$ halts. The essential undecidability remains because the introduction of unfixable states does not provide a mechanism for determining in finite time whether the machine is stuck indefinitely in a non-halting computation or will eventually halt.

Thus, the Halting Problem remains undecidable for TM^*, even with the presence of the fix function and unfixable states. The fix function may provide temporary resolution, but it does not fundamentally resolve the problem of undecidability:

$$H(TM^*, w) \text{ is undecidable.}$$

In conclusion, the extension of TM^* with unfixable states, fix functions, and fix states introduces additional computational paths but does not alter the inherent undecidability of the Halting Problem. While TM^* can attempt to correct errors via the fix function, it still cannot decide the halting status of all Turing machines and inputs.

5 Fault Tolerance and Probability of Failure

Fault tolerance is the capacity of a machine system to continue functioning effectively despite the failure of one or more components. We argue that the inherent limitations of machine systems imply that perfect fault tolerance is unattainable.

To analyze the operational reliability of a machine M, we define a probability function $P(W)$ that quantifies the probability of the machine being in a working state, incorporating time and space complexities:

$$P(W) = f(T(M), S(M))$$

where: - $T(M)$ denotes the time complexity for executing tasks on machine M, - $S(M)$ represents the space complexity required for the operation of M.

We express the probability of failure $P(F)$ as the complementary function of $P(W)$:

$$P(F) = 1 - P(W)$$

5.1 Graphical Representation

The graphical representation of $P(F)$ can be visualized in a two-dimensional space where: - The x-axis corresponds to time complexity T, - The y-axis corresponds to space complexity S, - The z-axis corresponds to the probability of failure $P(F)$.

Given a specific functional form for $P(W)$, we can denote:

$$P(W) = e^{-aT} \cdot \frac{1}{S^b}$$

where a and b are positive constants. Consequently, we have:

$$P(F) = 1 - e^{-aT} \cdot \frac{1}{S^b}$$

The probability of failure $P(F)$ exhibits a vertical asymptote when S approaches zero, which is interpreted as the scenario where insufficient space leads to failure. Specifically, when the machine encounters a halting problem, the behavior can be depicted as follows:

$$\lim_{S \to 0} P(F) = 1$$

This indicates that as space complexity diminishes, the probability of failure approaches certainty.

Figure 1: Graph of Probability of Failure $P(F)$ against Time Complexity T and Space Complexity S

5.2 Example Involving the Halting Problem

Consider a Turing machine TM^* that processes inputs leading to the halting problem. For specific inputs associated with the halting problem, such as an input sequence that results in an infinite loop, the machine will exhibit undefined behavior. As a result, we represent the function as:

$$P(W) = 0 \quad \text{when encountering halting problems}$$

Thus, we conclude that:

$$P(F) = 1 \quad \text{for halting problem inputs}$$

This signifies that for inputs that lead to halting issues, the machine cannot operate successfully, illustrating a fundamental limitation in machine capabilities.

5.3 Conclusion

The refinement of $P(W)$ into a function $f(T, S)$ provides a rigorous framework for assessing the reliability of machine systems. The graphical representation illustrates how the probability of failure escalates as time complexity increases and space complexity approaches critical limits. In the context of Turing machines like TM^*, the unavoidable presence of the halting problem accentuates these inherent limitations, necessitating oversight and intervention to navigate scenarios where automated systems falter.

6 Conclusion

The limitations of machines are inherently tied to the limitations of the formal systems S_H established by their human creators. Using ideas from Gödel's incompleteness theorem, we argue that no machine can fully repair itself in all cases, because there will always be states beyond the capabilities of the machine's formal system. In such cases, an external operator—the human H—must intervene, reinforcing the conclusion that machines can never be fully autonomous or self-sustaining without human-like intervention.