Halting Problem & Sanskrit Based On-board Supercomputer

"Inception vs perception" problem (solved in Sanskrit grammar), end of 1850s' black body radiation problem (still unsolved actually) and 1950s' halting problem are the same.

Halting problem can, in practice, be understood by:
- the failure to avoid queue in parallelism
- the inability of goal-oriented selection using machine learning

This affects quantum computers also.

Details:

The role of black body radiation studies in the end of 1850s drove progress in modern physics, due to initial difficulties in explaining the observed phenomena.

The Heisenberg Uncertainty Principle, which states that one cannot accurately measure both the position and speed of a particle simultaneously, has led to conventional quantum theories like the (1960s' version of) "S-matrix bootstrap theory" with still unsolved equations.

John Von Neumann, the physicist who made significant contributions to computer architecture and programming, offered key insights into the Halting Problem.

This problem, along with other aspects of quantum physics (as per Von Neumann, Weber, and others), is a part of several open-ended questions.

Quantum computers aim to address the perfect parallelism from a unique perspective, but they also face limitations rooted in this problem.

I offer to summarize the disputed and unproven aspects of quantum computers in both theory and application as follows:

1. Hadrons, as self-creating particles, present challenges, and conclusions on the data from the Large Hadron Collider (LHC) are debatable. Quarks and Gluons do not exist as per different theories and the indirect proofs being given for these particles can be explained using different theories also, not just the most popular one.

2. Existing practical applications like laser technology and quantum cryptography rely on 'conventional' quantum physics, despite unsolvable equations.

My intention of following questions is to stimulate thought and encourage contemplation.

Question 1:

I humbly request to see if there is any practical example of a parallel program, language, or software/hardware architecture that operates efficiently without relying on (algorithmic) queue. The utilization of queue imposes limitations on system parallelism and also demonstrates shortcomings in the algorithm.

For 1st question, examine these:

Bitcoin's no-queue causing reliability issue; Uber-like apps' billing delays due to queuing; Memory limits in queue causing message loss in message queues; Databases recommend disabling OpLocks at file write caching level to address inconsistencies; Async protocols for WAN performance utilizing hidden queues; Mutex implementations using queues; Itanium's failure for compiler-handled concurrency; Erlang parallelism used in WhatsApp being limited by queue memory; asynchronous circuits being limited by memory of the queue (analogous to queue is different here); concurrency handling issues in Domain Specific Languages; inevitability to increase out-of-order queue size in practice for superscalar processor designs; queuing limits in real applications with GPU programming, and the list can go on and on.

"As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality" - Einstein

Humbly request utile real-world examples only to challenge my views.

Question 2:

Could you please provide evidence of the functionality and effectiveness of goal-oriented selection using machine learning? I appreciate any context or resources you can provide, such as the article available at https://ncbi.nlm.nih.gov/pmc/articles/PMC4186236/

In all modesty, the complexity of the second question goes beyond this relatively short format. To summarize consider this evolution:

Skinner's work with pigeons and boxes, James Olds' rat experiments with those boxes, and the discovery of extraordinary pleasure centers that override basic animal instincts; triggering of unethical human brain experiments thereafter prompting a comprehensive ban; the 2000 Nobel Prize-winning discovery of non-invasive brain study techniques rekindling interest in brain research; resulting 2003 'BRAIN Initiative' and the concurrent rise of machine learning.

Many traditional Indian philosophers have deliberated on goal-oriented selection, opening up another complex discussion layer.

Solution:

Modern computers heavily rely on extensive optimizations at all levels to ensure usability. Systematic optimization is crucial to sustain progress and avoid issues like the problem of ending strings with zero, which persists across various levels of computing (https://queue.acm.org/detail.cfm?id=2010365). Notably, a grammar rule in Sanskrit prohibits statements from ending with zero or "sunya". Although run-length encoding is used in software and hardware as a solution these days in various contexts, the foundational problems and solutions are better explained in Sanskrit grammar.

By leveraging Sanskrit, we harness time-tested optimizations ingrained in the grammar itself, derived from the few foundational Maheswara sutras.

A time line of formal computer language grammars:
https://jeffreykegler.github.io/personal/timeline_v3

Utilizing Panini's Sanskrit grammar rules and mimansa, I believe it is possible to construct a hardware processor and directly employ Sanskrit as a programming language. This approach facilitates the creation of a supercomputing system that excels in all aspects.

Building upon existing research, my humble approach expands and enhances the context of Sanskrit in computer applications. I aim to demonstrate the following: In mimansa, there is a well-known situation where a servant, invited by a non-friend of the master (mother as per some texts) during an ongoing lawsuit, encounters the person on the road. The master then cryptically says, "eat the poison and die," meaning they intend to change the servant's mind. Traditional computers, regardless of model or programming, cannot guarantee reaching this conclusion. However, with Sanskrit, we can confidently arrive at the precise conclusion, demonstrating the steps on paper in a concise manner.

Importantly, all necessary texts on these subjects remain intact and available to this day, showcasing their enduring value over time. Specifically, Sanskrit's "Viswamitra rule" solves "inception vs perception" and the implications exceed common understanding.

Notes:

1. The omission of Alan Turing's interpretation is due to the intricate discussion on Mimansa (particularly Uttara Mimansa) his interpretation would necessitate. Given the challenge of simplifying Mimansa, I've respectfully sidestepped it.

2. The connection between physics and computer science becomes more apparent when examining the field of regex indexing.

3. Envision an operator like '*' that signifies combining of wave and particle nature when applied to numbers resulting in, 2 * 3 = 1. This might seem unconventional, yet I can validate it. I've encountered at least four pieces of work that reference this in the most enigmatic way conceivable.
Ref:
- <deleted in current version; details will be provided under NDA>
- <deleted in current version; details will be provided under NDA >
- <deleted in current version; details will be provided under NDA>
- Panini's grammar

As I delve deeper, these findings will undoubtedly lead to unprecedented insights about numbers that likely surpass any current imaginations. It's worth noting that this exploration will inevitably touch upon the realm of cryptography.

Epilogue:
There are way many things and what we touched is not even the tip of the iceberg. Each concept itself requires a book for general audience. But for those - well versed with any one of these 3: halting problem, quantum algebra, Sanskrit grammar - I am sure the understanding has come by now. Sanskrit grammar is all about wave-particle interplay. Thanks.

Revision History:
2019Aug17: Initial version
2025Jul26: Current version
2025Nov29: Added a line on run-length encoding, link to time line of grammar, epilogue.