Patent Title:

Computational Methods Driven by a Sanskrit-Based, Self-Refining Wave-Particle Numeric Framework

Abstract:

A computational method and system for performing dual-nature arithmetic operations derived from Sanskrit grammar. A processor receives two numeric operands and applies a Sanskrit-derived wave-particle operator "*" that fuses discrete (particle-like) and continuous (wave-like) semantic attributes. The resulting composite metric is then validated by cross-referencing at least four independent prior works, including the Viswamitra Rule of Panini's grammar and four confidential references, full disclosures of which will be provided to the Examiner in a confidential appendix in accordance with the Patent Office's confidentiality provisions. In one embodiment, for any two positive integers m and n, the system—based on this Sanskrit-based wave-particle computation—yields a quantum-semantic result, as exemplified by 2 * 3 = 1 in decimal base. This technology enables novel numeric transforms with applications in cryptography, semantic data analysis, and advanced, next-generation computing architectures.

The present invention further provides a computing system and method that leverage the formal grammar rules of Sanskrit—specifically Panini's Maheshwara Sutras and Mimamsa principles—to carry out computation, in some embodiments directly in hardware, thereby eliminating conventional queuing mechanisms and enhancing parallelism. The system inherently enforces constraints via the Viswamitra Rule, a deterministic grammatical guard derived from Sanskrit which guarantees determinism and improves goal-oriented selection. This architecture offers a self-optimizing supercomputer capable of real-time inference without reliance on traditional queue-based synchronization, with applicability to quantum computing, parallel software frameworks, and AI systems.

Description:

Field of the Invention:

This invention relates to computer-implemented arithmetic operations and, more particularly, to methods for performing dual-nature wave-particle numeric compositions inspired by Sanskrit grammatical principles. This invention further relates generally to computer architecture, parallel computation and machine learning & inference and in particular to the Halting Problem and Quantum Computing. It concerns a hardware-based supercomputing system that implements Sanskrit grammar as its native computation to achieve traditional queue-free parallel execution, goal-oriented selection processing for machine learning and inference. The system further solves quantum-algebraic equations by identifying the core computational problem embodied in an equation, mapping that problem to equivalent Sanskrit-grammatical constructs, and proving their equivalence to resolve the quantum-algebraic computation.

Background of the Invention:

Conventional arithmetic operators treat numbers as purely scalar entities, limiting their ability to capture richer semantic or contextual information. Emerging fields—such as quantum semantics, advanced cryptographic transforms, and cognitive computing—demand arithmetic primitives that fuse discrete and continuous attributes of numeric data. Applicant has discovered that modeling numeric operands through analogies to Sanskrit grammar, specifically leveraging Panini’s rules and the Viswamitra Constraint, yields a novel "wave-particle" operator. Unlike standard multiplication or addition, this operator outputs values encoding a composite metric, opening new avenues for semantic number theory and secure data transformations. Further, the traditional computing architectures rely on algorithmic queue-based synchronization to manage parallel tasks. Such queuing introduces latency and memory overhead—factors intimately tied to the Halting Problem. In contrast, Applicant has discovered what is herein called the Viswamitra Rule, a Sanskrit-grammar-derived constraint that ensures every computation path has determinism. Halting Problem which also comes in the way of goal oriented selection using machine learning and this again comes in the

way of Quantum computers similarly face constraints rooted in unresolvable algorithmic constructs.

Panini’s Maheshwara Sutras codify Sanskrit grammar to forbid certain constructs and prevent premature terminations. For instance, one derived rule bars any sentence from ending in sunya (zero), a constraint our system directly leverages. When this grammar-inspired framework is mapped onto machine operations, the resulting architecture obviates conventional queue mechanisms and enforces fully deterministic execution without auxiliary control structures. By contrast, existing computing architectures often introduce premature optimizations that rely on zero-terminated strings, a practice that continues to produce string-handling errors and related failures throughout the software and hardware stack. Sanskrit grammar based computational system guarantees no pre-mature optimization.

Definitions (for this disclosure):

As used herein:

Self-refining (also referred to herein as "self-optimization") :- The term "self-optimization" refers to the suite of optimization mechanisms native to Sanskrit grammar — embodied in Panini’s Maheshwara Sutras and Mimamsa principles — that enable the system to iteratively refine operand mappings and computational pathways without external tuning.

Person having ordinary skill in the art (PHOSITA):- Because this invention integrates Sanskrit grammar, theoretical computer science, the Halting Problem, and quantum algebra, a PHOSITA is assumed to have at least the following:
1. An understanding of the Halting Problem, as exemplified by the challenges of unavoidable queues in parallelization and goal-oriented selection set forth in Appendix A (Questions 1 and 2) of this specification;
2. A graduate-level grounding in compiler design and formal language theory, including their historical linkage to Sanskrit parsing;
3. Proficiency in Panini’s Maheshwara Sutras — particularly a deep understanding of the "why" aspect of the Visvamitra Rule — combined with Mimamsa inference principles, and their similarities to modern semantic-computing approaches (e.g., machine learning);

4. Knowledge of quantum-algebraic methods and, importantly, their limitations.

Summary of the Invention:

The present invention comprises four primary modules—Core Module (CM), Translation Module (TM), Programming Module (PM), and Pre-Programming Module (PPM)—each of which may be implemented in software, hardware (e.g., general-purpose processors, FPGAs, or ASICs), or a combination thereof, and which collectively enable Sanskrit-based dual-nature arithmetic computation.

Core Module (CM):-

Implements the processor-based technique for a Sanskrit-derived wave-particle arithmetic operation:

1.1. Operand Acquisition

Receive a first operand and a second operand.

1.2. Dual-Nature Operation

Apply a "*" operator that fuses discrete (particle-like) and continuous (wave-like) semantic attributes of the operands.

1.3. Result Generation

Produce a result value encoding the composite wave-particle metric.

1.4. Result Validation:

Cross-verify the result against (i) the Viswamitra Rule of Panini’s grammar and (ii) four confidential references, full disclosures of which will be provided in a confidential appendix to the Examiner under the Patent Office’s confidentiality provisions.

Translation Module (TM):-

Maps an arbitrary, high-level problem statement into operands and invokes the CM; comprises:

2.1. Wave-Component Extraction

2.2. Particle-Component Extraction

2.3. Core Dual-Nature Invocation (as per CM steps 1.2-1.4)

2.4. Iterative Refinement

- If validation fails, return to Wave-Component Extraction.

- Upon successful validation, record the final operand mapping for developer reference to improve subsequent TM versions.

- If validation continues to fail after n iterations, log the full trace to aid debugging and CM self-optimization.

Both the CM and TM incorporate a Sanskrit Grammar & Inference Engine that parses input strings using Panini's Maheshwara Sutras and Mimamsa principles, thereby eliminating algorithmic queues in parallel execution and enforcing deterministic, goal-oriented branch selection.

Programming Module (PM):-
A repository of text files containing natural Sanskrit sentences, for direct input to the TM.

Pre-Programming Module (PPM):-
Processes an arbitrary collection of text files written in any natural language to produce a valid PM.

Example:- A developer writes using Sanskrit-based constructs and executes them via the PM, while a general user's input is handled through the PPM.

Claims:

What is claimed is:

1. A computational method for performing a dual-nature arithmetic operation, the method comprising:
a. receiving, at a processor, a first operand and a second operand;
b. applying, by the processor, a Sanskrit-derived wave-particle operator "⋆" to the first and second operands, wherein the operator "⋆" is configured to fuse discrete (particle-like) and continuous (wave-like) semantic attributes of the operands into a composite wave-particle metric;
c. generating, by the processor, a result value encoding the composite wave-particle metric; and
d. providing the result value.

2.The method of claim 1, wherein the generating of the result value further comprises validating the result by cross-referencing at least five independent prior works, the cross-referencing including:
(i) the Viswamitra Rule of Panini’s grammar; and
(ii) at least four confidential references, full disclosures of which will be provided to the Examiner in a confidential appendix in accordance with the Patent Office’s confidentiality provisions.

3. A Sanskrit-based self-optimizing on-board supercomputer system, comprising:
(a) a Sanskrit Grammar Module configured to parse input instructions according to Panini’s Maheshwara Sutras and Mimamsa principles;
(b) a Mimamsa Inference Engine coupled to the Sanskrit Grammar Module and configured to dynamically select computational branches based on context and goal-oriented evaluation without iterative search;
(c) a Compiler configured to translate high-level problem descriptions into sequences of Sanskrit grammar instructions;
(d) a Microarchitectural Mapping Layer that maps parsed Sanskrit instructions directly to processing elements and eliminates algorithmic queues during parallel execution;
(e) a plurality of Processing Elements operable in parallel to execute the mapped instructions; and
(f) the Viswamitra Rule Module of claim 1;

wherein the system enforces deterministic processing and obviates conventional queue-based synchronization mechanisms.

4. The system of claim 3, wherein the Sanskrit Grammar Module implements all instructions as direct applications of Panini’s sutras or Mimamsa inference rules.

5. The system of claim 3, wherein the Mimamsa Inference Engine enforces goal-oriented selection by applying Mimamsa’s principles to resolve branching without requiring post-compilation searches.

6. The system of claim 3, wherein the Microarchitectural Mapping Layer prevents any algorithmic queueing exclusively by virtue of the Sanskrit instruction-based mapping.

7. The system of claim 3, wherein the plurality of Processing Elements are implemented on one or more field-programmable gate arrays (FPGAs).

8. The system of claim 3, further comprising a Quantum Algebraic Solver Module that maps Sanskrit grammar constructs to quantum algebraic formulations for solving S-matrix bootstrap equations.

9. The system of claim 8, wherein the Quantum Algebraic Solver Module is configured for real-time inference in quantum computing and artificial-intelligence applications.

10. The system of claim 3, wherein the Compiler further supports translation of non-Sanskrit source code into Sanskrit instruction sequences.

11. A method for on-board execution of quantum algebraic equation solving in a Sanskrit-based self-optimizing supercomputer, comprising:

(a) translating a high-level problem description into a sequence of Sanskrit grammar instructions using a compiler;

(b) parsing the Sanskrit instruction sequence via a grammar module that enforces Panini’s Maheshwara Sutras and Mimamsa rules;

(c) mapping each parsed instruction directly to operations on processing elements via a microarchitectural mapping layer, thereby eliminating algorithmic queues;

(d) executing the mapped instructions in parallel on the processing elements; and

(e) applying a Mimamsa inference process to select computational branches in real time without iterative search.

12. The method of claim 11, wherein the parsing step prevents constructs equivalent to "sunya" endings in instruction strings, thereby guaranteeing non-premature termination.

13. The method of claim 11, further comprising mapping selected Sanskrit instructions to quantum algebraic operations to solve S-matrix bootstrap equations.

14. The method of claim 11, wherein the execution step is performed on one or more FPGAs implementing the microarchitectural mapping layer.

15. The method of claim 11, further comprising translating non-Sanskrit code into Sanskrit instructions that correspond to Mimamsa inference rules for goal-oriented computation.

16. A computer program product, comprising a non-transitory storage medium bearing program code that, when executed by one or more processors, causes the system to perform the method of any one of claims 11 to 15.

Incorporation by Reference:

Appendix A, entitled "Sanskrit-Based On-Board Supercomputer" and "Sanskrit Technologies: 1st POC," is hereby incorporated by reference in its entirety.

Appendix A: Combined Document Follows from Next Page

Sanskrit Technologies: 1st POC

1. We use an fpga using a traditional algorithm and solve a problem like shortest path.

2. We use an fpga using our Sanskrit grammar based algorithm and solve the same problem. In doing so, we will show leapfrog improvement in performance.

All we need is to write the code in C with cloud instances supporting fpga.

3. Fpga:

    3.1. We shall use vitis_hls and get the .xo file for, say, the sanskrit.c

    3.2. Using AWS HDK bash script, get flash image and publish the image

    3.3. Load this in AWS F1 instance

    3.4. Write run.c code and using mmap run the fpga.

4. To write the said sanskrit.c, estimated time is, maximum 4 years.

Document History:

Bangalore, 2025Jul06: GNVS Sudhakar: Initial document
Bangalore, 2025Jul26: GNVS Sudhakar: Current version

Ancient Samskrt Language and Modern Computers

Seminar by,
Sudhakar, GNVS.
V MCA ,
August 2000

Seminar Guides:

Dr.S.M. Hegde,
Asst. Professor,
Dept. of MACS

and

Mr. D.Pushparaj Shetty,
Lecturer,
Dept. of MACS

Karnataka Regional Engineering College
Surathkal