GOODREASON RESEARCH WHITEPAPER

On Symbolically Computable Systems

Representation, analysis, reasoning and execution under partial knowledge

α π χ ΔΨ β φ τ Ω

Eki Laitila, PhD

Independent systems scientist | GoodReason

Version 1.0 | 21 August 2026

Suomenkielinen saate

Tämä whitepaper määrittelee symbolisesti laskettavan systeemin käsitteen. Lähtökohtana on neljä toisiaan täydentävää vaihetta: symbolinen esittäminen, symbolinen analyysi, symbolinen päättely ja symbolinen suorittaminen. Käsitteen taustalla ovat tekijän vuonna 2008 valmistunut symbolisen analyysin väitöskirja, pitkä Visual Prolog -kehitystyö sekä GoodReasonin α-Ω-koordinaatisto.

Computation tarkoittaa tässä enemmän kuin numeerista laskentaa: symbolien esittämistä, tyypittämistä, sidontaa, evaluointia, päättelyä, simulointia ja systeemitilan muuttamista. Muuttuja voi olla vapaa, kysymys voi säilyä avoimena ja epädeterministinen prosessi voi tuottaa useita vaihtoehtoisia sidontoja. Jokainen tulos säilyttää perustelunsa, lähteensä ja voimassaoloehtonsa, joten myös ratkaisematon kysymys on käsiteltävä tieto-objekti.

Symbolinen laskettavuus koskee rajattua systeemimallia, jonka käsitteet, säännöt ja suoritusehdot on tehty näkyviksi. Se ei suurenna Churchin ja Turingin määrittelemää matemaattisen laskettavuuden luokkaa, vaan laajentaa computation-käsitteen systeemiseksi ja semanttiseksi käytännöksi, jossa ihminen määrittelee tarkoituksia ja voi tarkastaa, kysyä ja korjata symbolisen prosessin etenemistä.

Versio 1.0 erottaa toisistaan laskettavuuden, todistettavuuden ja käytännöllisen ratkaistavuuden. Looginen todistus on aina suhteessa aksioomiin, semantiikkaan ja esitettyyn kontekstiin. Moduulin tarkoitettu input-output-suhde kuuluu sen määriteltyyn logiikkaan, mutta sivuvaikutusten hallinta on sisällytettävä mallin ehtoihin. NP-kompleksisuus puolestaan kuvaa resurssien kasvua ja todistusten löytämisen vaikeutta, ei yksin sitä, onko ongelma periaatteessa laskettavissa.

Ydinkysymys: Voidaanko systeemiä käsitellä symbolisesti niin, että tarkoitukset, suhteet, avoimet muuttujat, vaihtoehdot ja perustelut säilyvät sekä ihmisen että koneen käytettävissä?

Abstract

Systems science has developed powerful concepts for boundary, purpose, hierarchy, feedback, emergence and change. Computational logic has developed

equally powerful mechanisms for variables, unification, inference, nondeterminism and execution. Yet these traditions are rarely integrated at the level where a system model can remain epistemically open and still be computationally active. This whitepaper introduces the concept of a symbolically computable system (SCS): an explicitly delimited system model whose relevant entities, relations, purposes, constraints, unresolved questions and evidence are represented by typed symbols, and whose knowledge state can be transformed through explicit rules without requiring every variable to be bound or every question to be decidable. The proposed capability chain distinguishes symbolic representation, symbolic analysis, symbolic reasoning and symbolic execution. A relational result semantics differentiates justified answers, refutation under declared assumptions, open questions and supported conflicts. The paper connects this proposal to Turing's account of computability, Robinson's resolution principle, Kowalski's separation of logic and control, classical symbolic execution, Visual Prolog flow semantics, multi-agent systems and systems-language research. It separates computability, formal provability and computational tractability; treats proofs as relative to declared semantics, context, side-effect conditions and resource bounds; and places computation itself at the center as representation, typing, substitution, binding, evaluation, inference, simulation and transformation. Justification traces provide an operational bridge to human metacognition and machine self-monitoring. GoodReason supplies a systemic coordinate space through eight perspectives (α-Ω) and seven epistemic rings. The Symbolic Abstract Machine supplies an implemented bridge from Turing's abstract account to typed semantic atoms and interpretable program behaviour. The resulting construction connects systems inquiry, symbolic AI and operational knowledge structures.

Keywords: symbolic computation; systems science; computational logic; symbolic analysis; logic programming; partial knowledge; four-valued information; abduction; computational complexity; metacognition; computational semiotics; multi-agent systems; GoodReason

Contents

1. Research problem and contribution

Systems inquiry often begins where complete specification ends. The system of interest may have contested boundaries, several purposes, changing actors, incomplete evidence and multiple plausible futures. Conventional computation, by contrast, is commonly presented as a mapping from specified inputs to determined outputs. This contrast can create the impression that formal logic is too narrow for systemic inquiry and that systems thinking must remain primarily descriptive or interpretive.

That conclusion follows only if logic is equated with a closed binary calculus. Logic programming provides a different starting point. A query may contain free variables; unification may bind them; nondeterministic search may return many admissible bindings; backtracking may restore an earlier state; and failure may be interpreted only relative to the declared knowledge and control regime. This allows computation to proceed without pretending that the model is complete.

> **Core thesis:** Logic does not reduce a system to certainty. It specifies how reasoning can proceed when knowledge is partial, alternatives are multiple and some questions must remain open.

The paper contributes six connected elements: a four-stage symbolic capability chain; an explicit definition of a symbolically computable system; a result space for

partial and conflicting knowledge; an interpretation through GoodReason's α-Ω geometry; the implemented Symbolic Abstract Machine; and a foundation for accountable human and multi-agent coordination. Together they define a working symbolic paradigm whose claims remain inspectable, revisable and bounded by explicit semantics and computational resources.

2. From computable numbers to computable system models

In **cognitive science**, computationalism and connectionism are two major models of the philosophy of mind. Computationalism sees *human as a symbolic system*. In stead, connectionism views mind as a neural network processing information through interconnected nodes, inspired by the structure of the brain [29].

2.1 Computability and symbols

Turing's 1936 paper defined computable numbers through finite symbol-manipulating procedures and established both the power and limits of mechanical computation [1]. The present proposal does not extend Turing computability by decree. It changes the object being modelled: from a number-producing machine to an explicitly bounded system representation whose states include claims, relations, questions, assumptions and justifications.

Church's lambda calculus and Turing's machines provided extensionally equivalent accounts of effective calculability in the same foundational period [21]. Symbolic logic itself has deeper roots in the work of Boole, Frege, Peano, Russell and others; Church's contribution was to make symbolic operation central to a precise theory of computability. The Chomsky hierarchy later organized generative grammars by the restrictions placed on their rules and related unrestricted Type-0 grammars to Turing-machine computation [28].

In this paper computation is therefore a verb: the rule-governed treatment of represented objects. It includes numerical calculation, but also typing, substitution, variable binding, evaluation, inference, simulation and transformation. The mathematical class of computable functions remains unchanged; the contribution concerns the meanings, systemic relations and human purposes made operational within that class.

Example. *What is Economy as a system? Hereafter, Economy denotes the selected System of Interest examined throughout the paper.*

Newell and Simon later framed intelligent action in terms of physical symbol systems and search [2]. Their hypothesis made symbols central to artificial intelligence, but the SCS proposal places equal emphasis on epistemic status. A symbol may denote what is known, what is assumed, what is sought, what is disputed or what remains outside the current model.

Stanford's Symbolic Systems program offers an important interdisciplinary comparison by joining computer science, linguistics, philosophy, psychology, mathematics and statistics around natural and artificial systems that represent, process and act on information [23]. SCS is not a curriculum; it is a computational and systemic construction that makes representation, analysis, reasoning and execution explicit within a single operational model.

2.2 Machine-oriented inference

Robinson's resolution principle provided a machine-oriented foundation for automated theorem proving [3]. Kowalski's formulation algorithm = logic + control then separated declarative knowledge from the strategy used to apply it [4]. This distinction is essential for systems science: two agents may share a logical knowledge model while employing different search, priority, risk or stopping policies.

Visual Prolog makes information flow explicit. Predicate flow patterns must match the free/bound status of arguments; unification binds variables while leaving as much as possible open; and a predicate may succeed more than once through backtracking [5]. These are not merely programming details. They provide an operational vocabulary for incomplete systemic knowledge.

2.3 Computability, provability and tractability

Three questions must be kept separate. Computability asks whether a procedure can in principle produce a result. Provability asks whether a proposition follows from declared axioms and inference rules. Tractability asks how time and memory grow with problem size. Cook's analysis of theorem-proving procedures established the foundation of NP-completeness by relating polynomially checkable certificates to problems whose solutions may be difficult to find [16]. The unresolved P versus

NP question therefore limits claims about efficient general solution, but NP membership is not a declaration that a problem is uncomputable.

A formal proof of program or system behavior is also relative to a specification. Floyd and Hoare showed how assertions, preconditions, postconditions and program semantics support proofs of correctness [17,18]. A module's intended input-output relation belongs to its declared logic and natural execution flow. If relevant context, shared-state interactions or side effects are omitted, a proof may be valid for the represented model while failing to establish the desired claim about the wider system. Local and separation-based reasoning make the boundary of affected state explicit rather than assuming that the untouched context is irrelevant [19].

Logical safeguard: A theorem is proved within a formal system and a declared model. The proof establishes the conclusion only under the represented assumptions, semantics, context and side-effect boundary; complexity determines whether finding or checking that proof is feasible at the required scale.

2.4 Symbolic analysis as an intermediate capability

Laitila's dissertation introduced symbolic analysis and an atomistic hybrid model for program comprehension [6]. Its architecture moved from GrammarWare to ModelWare, SimulationWare and KnowledgeWare. Source programs were transformed into symbolic atoms combining object-based encapsulation with logic-language expressiveness; symbolic simulation then supported analysis and knowledge capture. The present paper generalizes that experience from program comprehension toward system comprehension.

The Symbolic Abstract Machine and its atomistic computation model were also documented in Symbolic Analysis as a Basis for Program Comprehension [12]. This continuity matters: the present construction extends an implemented line of symbolic simulation and program comprehension rather than introducing symbolic computation as a future possibility.

Symbolic analysis is therefore not a synonym for reasoning. It is the interpretive and structural work through which representations are inspected, decomposed, connected, compared and prepared for inference. A human, a machine or a mixed human-machine process may perform it.

3. The four-stage symbolic capability chain

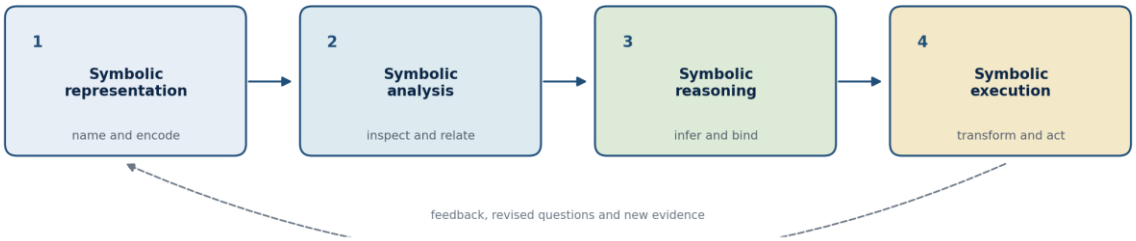

Figure 1. Four symbolic capabilities with a feedback path from execution to revised representation.

Stage	Primary question	Characteristic result
1. Representation	What is named and encoded?	Typed symbols, relations, states and identifiers
2. Analysis	What structure and meaning can be found?	Decomposition, links, patterns, anomalies and possible semantics
3. Reasoning	What follows, under which assumptions?	Bindings, alternatives, refutations, open questions and reasons
4. Execution	What symbolic state is transformed or enacted?	Updated knowledge, selected action, simulation trace or agent commitment

3.1 Symbolic representation

Representation names and encodes selected aspects of a system: entities, relationships, purposes, boundaries, contexts, events and states. It is comparable to coding in the broad sense, but it is not yet analysis. A representation is always selective. It should therefore record its SOI boundary, intended use, author or agent, version and evidential status.

A semiotic reading clarifies what representation preserves. In Peircean terms, a sign refers to an object through an interpretant: the developed understanding or further sign through which the relation becomes meaningful [20]. An identifier, diagram or traffic signal is therefore not self-explanatory. Its meaning depends on a convention, interpreter and context of use.

Computational semiotics joins this account of signs and meaning with computer science, artificial intelligence and formal modelling [13]. Semiotics contributes

object-sign-interpretant relations and contextual meaning; computation contributes executable formalisms through which such relations can be modelled, tested and transformed. Symbol is consequently not a narrow content type below text or images, but the organizing relation through which texts, images, models and actions acquire meaning.

3.2 Symbolic analysis

Analysis turns a representation into an inspectable model. It may identify dependencies, control paths, recursive structures, missing links, incompatible types, repeated patterns or semantic roles. In the original program-comprehension work, symbolic execution of an atomistic model exposed program behavior. In a broader SCS, analysis may also compare theories, actor accounts and alternative system boundaries. **Example**. How do we analyze Economy?

Humans and machines may implement this interpretive step differently. A person can recognize a red traffic light as a socially learned sign and relate it to the present traffic situation. A machine can parse a typed field, resolve an identifier or match a symbolic structure against a schema. In both cases analysis establishes relations that reasoning or action can subsequently use.

3.3 Symbolic reasoning

Reasoning applies declared inference rules, constraints and control policies. It can bind free variables, test already bound values, generate alternatives, propagate consequences and retain justification paths. The aim is not necessarily a single answer. A valuable result may be a set of admissible alternatives or a precise description of why the question remains open.

3.4 Symbolic execution

In classical computer science, symbolic execution runs a program with symbolic rather than concrete inputs and accumulates path conditions [7]. This paper preserves that established meaning while defining a broader systemic use. SCS execution transforms a symbolic system state according to an explicit execution semantics. The transformation may simulate a process, activate an inference object, revise a knowledge state, generate an agent commitment or issue an externally executable action. Classical program-path symbolic execution is one special case. In software testing it can explore feasible paths and generate inputs

for branches or states that concrete tests have not yet reached; coverage is an evaluation objective, not a synonym for symbolic execution [7].

Terminological safeguard: Representation describes; analysis reveals; reasoning derives; execution changes or enacts a symbolic state. A system may support one stage without supporting all four.

4. Definition of a symbolically computable system

Definition 1: A symbolically computable system is an explicitly delimited system model in which relevant entities, relations, purposes, constraints, unresolved questions and evidence are represented by typed symbols, and in which rule-governed operations can generate and validate justified successor knowledge states without requiring every variable to be bound or every question to be decidable.

The word system refers here to a system model relative to a selected SOI and MOI, not to an ontological assertion that the entire real-world system is algorithmically computable. The phrase symbolically computable identifies a capability of the representation-inference-execution arrangement. Model validity remains empirical, interpretive and purpose-dependent.

The System of Interest (SOI) identifies what is being examined. The Meaning of Interest (MOI) identifies the purpose-dependent interpretation through which a human, community or agent understands that system. MOI is not a static label: it may evolve as symbolic analysis, reasoning, execution and new evidence transform the knowledge state.

$$SCS = (SOI, MOI, \Sigma, T, K, R, C, J, A, E)$$

Element	Meaning
SOI	Declared system of interest, boundary and environment
MOI	Purpose-dependent meaning, viewpoint and evolving interpretation of the SOI
Σ	Symbol vocabulary, identifiers and address space
T	Types, domains and admissible structural forms
K	Partial knowledge state: claims, questions, evidence and statuses

Element	Meaning
R	Inference and transformation relations
C	Constraints and control policies
J	Justification and provenance structures
A	Human or artificial agents and their local knowledge states
E	Execution semantics connecting symbolic transitions to simulations or actions

4.1 Actor-relative and versioned meaning

In a multi-agent setting, MOI should be represented rather than assumed. Agent a_j may hold an interpretation m_j containing its viewpoint, purpose, present interpretation, provenance and version. New evidence or a new justification may revise that state without silently overwriting another agent's interpretation.

$$m_j = (SOI, viewpoint_j, purpose_j, interpretation_j, provenance_j, version_j)$$
$$MOI_j(t+1) = revise(MOI_j(t), \Delta K_j, J_j, C_j)$$

4.2 Explicit logic and accountable boundaries

An SCS operates only on distinctions that have been represented in its current model. Its symbols, types, rules, validity conditions and stopping policies are available for inspection, and its claims are limited to the declared SOI, MOI and evidence boundary. Unknown, underdetermined and random are therefore different model statuses: each must be stated together with the grounds on which it is used.

This explicitness protects both logic and systemic inquiry. Formal evaluation can be exact within its represented domain, while the boundary records what has not been modelled. New observations or interpretations may extend the model without being retroactively treated as premises of an earlier proof.

A statistical language model may assist symbolic analysis by proposing interpretations or possible relations. Its output enters K as a sourced contribution; the SCS layer retains explicit types, constraints, provenance, result status and control. A fuller treatment of intelligence, consciousness and higher-level agent cognition is reserved for a companion paper.

Example. Are we getting enough information about Economy?

5. Operational semantics for partial knowledge

5.1 A relational result space

Because reasoning may be nondeterministic, the evaluation of a query is better represented as a relation than as a single-valued function. Let K be a partial knowledge state and q a query. The result relation may yield one or more justified successor states or an explicit epistemic status.

Result(K,q) ∈ { answer(theta_i, M_i, K_i_next), refuted(q,M), open(q,C,M), conflict(q,{M_pos},{M_neg}) }

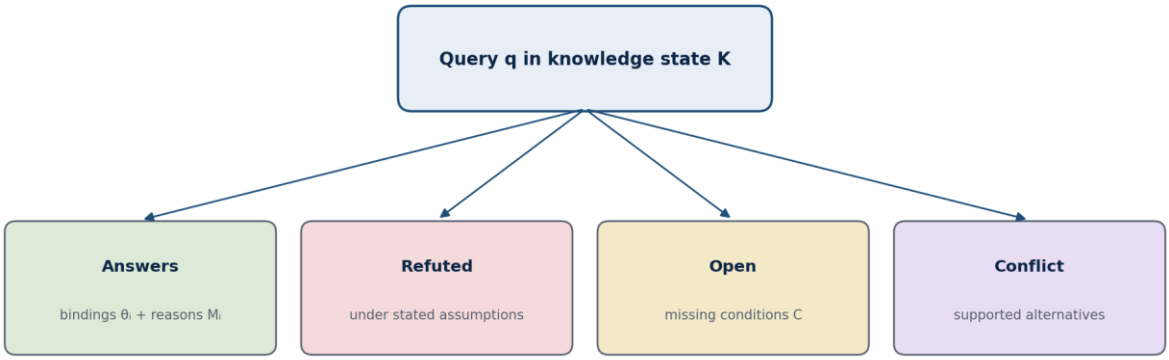

Figure 2. The result of systemic reasoning is not restricted to one answer or binary failure.

An answer contains a substitution θ_i, a justification M_i and a successor knowledge state K'_i. Refutation is valid only under declared assumptions and a specified logical regime. Open means that the question is retained together with missing conditions C and the reason M for its status. Conflict preserves supported positive and negative positions instead of silently overwriting one.

5.2 Neighbouring formalisms: four values and abduction

Belnap's four-valued logic distinguishes information that is supported as true, supported as false, supported both ways or unsupported either way [26]. This is a close neighbour of answer, refuted, conflict and open, but the SCS result objects additionally carry substitutions, conditions, provenance, justifications and successor states. The correspondence is therefore a useful positioning device, not an identity between formalisms.

Abductive logic programming offers another close relation: it searches for hypotheses that, together with a program, explain an observation while satisfying integrity constraints [27]. The missing conditions C in open(q,C,M) may identify candidates for later abduction, but they are not automatically admissible hypotheses. Their status must be declared and constrained.

5.3 Free and bound variables

A variable is free relative to a particular inference state when no value has yet been assigned within that state. It may become bound through unification. On backtracking, the binding can be withdrawn and another admissible value generated. Thus one query may produce a very large set of bound substitutions without implying that the variable possessed all those values at once. Visual Prolog's flow patterns and reference-variable tests make this transition operationally visible [5].

The distinction is epistemically useful. A free variable is not an error or an empty database field. It represents an explicitly open position in a relation. A bound value is not necessarily final truth; it is a value supported in a particular solution state under specified assumptions.

5.4 Circular relations, non-termination and the question as knowledge

$$A = f(Q) \qquad Q = g(A)$$

$$M = \text{justification of the relation and its limits}$$

Circularity may indicate recursion, feedback, mutual definition, an unresolved constraint cycle or an error. An SCS must not decide among these interpretations merely from the shape of the equations. It may seek a justified fixed point, generate several consistent solutions, detect non-termination or preserve the cycle as an open constraint object.

Computability boundary: If SCS execution can simulate arbitrary Turing machines, no universal procedure can decide for every SCS query whether evaluation will terminate or reach a final resolution [1]. A visible cycle alone does not establish this impossibility; the boundary follows from the expressive power of the execution semantics.

Example: Do Economy's feedback loops work to promote equilibrium?

This gives a precise interpretation to the proposition that, when the answer is not known, the question itself may be the best available knowledge. The query is retained as open(q,C,M): it has identity, context, missing conditions, provenance and a route for later reactivation. It is not falsely promoted into a resolved answer. When a resource or stopping policy is exhausted, open(q,C,M) records unresolved status and its operational reason; it does not by itself prove that the query is mathematically undecidable.

5.5 Justification as a first-class object

Every consequential binding or state transition should be accompanied by a justification object M. The object may reference a rule, source, observation, agent, timestamp, confidence assessment, counterargument and validity scope. Argumentation frameworks offer a formal basis for representing attacks and defenses among arguments [9].

An individual justification M is an instance within the broader justification structure J. For a human reasoner, accumulated M objects can support metacognition: knowledge and monitoring of one's own cognitive tasks, strategies and results [24]. In GoodReason this bridge is visible at π4, where reasoning becomes an object of reflection, and χ4, where that reflection becomes communicable information. For a machine, a trace, proof object, diagnostic record or confidence estimate is not evidence of consciousness. It is an operational analogue: the system can inspect records of its own inference and use them to revise control, request evidence or reopen a query [25].

6. GoodReason as a systemic coordinate space

GoodReason supplies an addressable geometry for distributing the elements of an SCS. Its eight perspectives are not inference rules by themselves. They organize what the inference concerns, thereby reducing category mistakes between purpose, theory, evidence, change, structure, design, implementation and evaluation [11].

This orientation is compatible with Capra and Luisi's systems view of life, which connects living, cognitive and social organization through recurring relational principles [22]. GoodReason adds an addressable symbolic form through which such multiple perspectives can be compared and revised without reducing the whole to one disciplinary vocabulary.

Example. Can Economy be seen as a coordinate system with SOI as its origo?

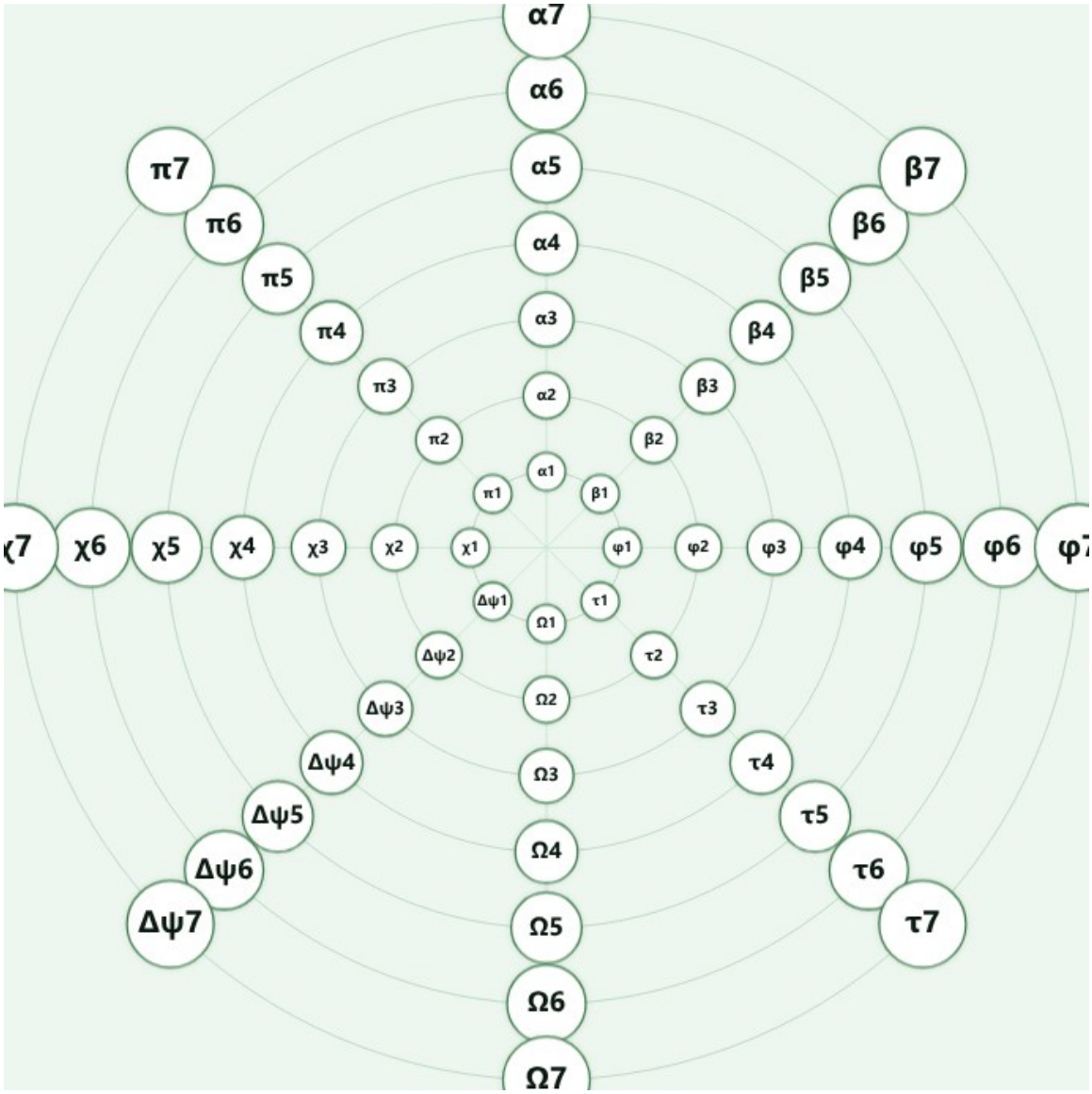

Figure 3a. The GoodReason coordinate space. The α–Ω axes retain the same directions in every system, enabling isomorphic comparison, a consistent human experience and alignment between software technology and systemic purpose.

Coordinate	Role in a symbolically computable system
α Purpose / mind	Declares SOI, MOI, question, intent and success orientation
π Theory / principle	Holds axioms, concepts, models, rules and theoretical commitments
χ Environment	Holds observations, sources, context, external conditions and evidence
ΔΨ Change pressure	Represents open variables, anomalies, tensions, transitions and reframing

Coordinate	Role in a symbolically computable system
β Organization / structure	Defines types, actors, relations, constraints and knowledge architecture
φ Design / solution	Generates and compares possible bindings, designs and interventions
τ Integration / implementation	Executes selected transformations, simulations, actions and interfaces
Ω Developmental arc / feedback	Validates outcomes, retains provenance, learns and reopens questions

The seven rings add epistemic depth: 1 Vakaus (stability), 2 Epävarmuus (uncertainty), 3 Kompleksisuus (complexity), 4 Herääminen (awakening), 5 Emergenssi (emergence), 6 Murros (transformation) and 7 Pohdinta (reflection). They should not be reduced to a linear difficulty score. A ring indicates the character of awareness and inquiry at a coordinate. For example, φ1 may contain a stable operational solution while φ7 examines what should count as a solution at all.

6.1 One systemic question across seven epistemic rings

A single deliberately open query—q_econ = 'What is an economic system?'—can expose the different contracts of inquiry. A bounded source may return a definition; a formal model may adapt it to declared inputs; different actors may supply incompatible purposes and boundaries; and a multi-agent inquiry may encounter both missing and excessive information. The wording remains fixed while the represented context, permissible operations and result status change.

One question across seven depths of inquiry

Ring	7C reference / CPSS view	Chomsky / language level	Language-model contribution	GoodReason systemic interpretation
1	Connection Connection to money	Type 3 / regular Earning	A prompt returns a bounded answer from statistical knowledge.	A minimal model of economic circulation at α1…Ω1: the smallest coherent description.
2	Conversion Exchange	Type 2 / context-free Profession	A situated description adapts the answer to declared circumstances.	A complicated declarative model records structure, exchange relations and configured meanings.
3	Cyber Boundedness	Type 1 / context-sensitive Workplace	A negotiated answer identifies constraints and competing interpretations.	A critical model of a complex economy makes feedback loops, boundaries and conflicting purposes visible.
4	**Communication Conflict of values**	**Type 0 / Turing Continuity and halting**	**Heterogeneous sources raise the question: who can be trusted, and under which interpretation?**	**Communication exposes different views in the economic system: Chaos A=f(Q), Q=g(A); the result may be answer, refuted, open or conflict.**
5	Cognition The economy is possible	—	Synthesis is useful, but unsupported completion and hallucination remain possible.	An inquiry loop expands evidence, theories and viewpoints when the earlier source is insufficient.
6	Configurator Creative monetary policy	—	Generation can support design, but cannot independently ground purposes, values or legitimate authority.	Models are reconfigured under explicit human purposes, constraints and accountable comparison of plural viewpoints.
7	**Collective Intelligence What should we want?**	**—**	**The model can formulate alternatives; responsibility for ideals, governance and choice remains human.**	**Inquiry asks what counts as knowledge and a desirable whole. A viable redesign begins from β and returns through Ω-feedback.**

Figure 3b. The same economic-system query is interpreted across 7C, the Chomsky hierarchy, language-model contribution and GoodReason. Ring 4 marks the watershed where unrestricted computation meets explicit open and conflicting knowledge states.

The rows are cross-model projections onto a shared coordinate of cognitive depth. They need not be identical objects: their differences are the information carried by the comparison. Chomsky's Type 3, Type 2, Type 1 and Type 0 classes describe progressively less restricted generative rules and corresponding machine models [28]. GoodReason asks how a question changes when bounded syntax, context, computation, meaning, values and systemic responsibility are brought into the same inquiry. The fourth ring is critical because unrestricted symbolic execution encounters possible non-termination, while inquiry must distinguish answer, refutation, openness and supported conflict.

At the outer rings, the inquiry can ask who defines knowledge, whose purposes govern the system and which ideals—such as sustainability—should guide redesign. Fragmentation does not decrease automatically. It may decrease when actor-specific MOIs, assumptions, evidence and conflicts are made inspectable and feedback supports coordinated revision without erasing legitimate difference.

The α-Ω coordinates organize systemic meaning; logic defines admissible inference; control selects a search strategy; execution changes the symbolic or external state; and Ω returns evidence to the next inquiry cycle. This recurrent movement is the systemic meaning of the coordinate space, not a linear ranking of subjects or people.

7. Multi-agent reasoning and collective intelligence

Example. Can Economy be explored by independent agents via messages (promt)?

A multi-agent SCS does not require all agents to share one complete worldview. Each agent a_j may maintain a local knowledge state K_j, a set of capabilities, priorities and commitments, and an explicit policy for accepting or rejecting information. Shoham's agent-oriented programming demonstrated how beliefs, capabilities, choices and commitments can be treated computationally [8].

$$a\_j = (MOI\_j, K\_j, R\_j, C\_j, goals\_j, commitments\_j, provenance\_j)$$

Agents exchange more than conclusions. A useful message contains the query or claim, variable bindings, assumptions, justification, provenance, confidence, requested action and validity interval. This allows another agent to reproduce, challenge, qualify or defer the conclusion. Disagreement becomes structured information rather than a communication failure.

Collective intelligence emerges when several agents can preserve their differences while still coordinating on shared symbols and transition rules. GoodReason's coordinates offer a common address space: one agent may contribute χ-evidence, another π-theory, another φ-design and a fourth Ω-validation. The architecture supports distributed reasoning without assuming that one agent owns the complete model.

Example. Can Economy be understood as an autonomous machine (SAM), or is it an implementer of the will of its environment: master or slave?

8. Implementation Philosophy of Computation

Turing's analysis of computation began from the activity of a human computer: a person manipulating written symbols according to explicit rules. The resulting abstract machine defined effective computation independently of any particular hardware. Relay machines, electronic computers, semiconductor technology and integrated microprocessors later transformed this principle into industrial infrastructure. In formal language theory, the correspondence between unrestricted Type-0 grammars and Turing machines established the computational boundary of the Chomsky hierarchy. The Turing machine, the Chomsky hierarchy and their physical implementations therefore represent related but distinct achievements: an abstract definition of computation, a hierarchy of generative languages and a technological architecture for executing programs.

The Symbolic Abstract Machine (SAM) developed in Laitila's research extends this lineage from tape-level symbols to programming-language semantics. It was implemented for the symbolic simulation and analysis of Java programs and documented in the 2008 dissertation Symbolic Analysis and Atomistic Model as a Basis for Program Comprehension. Chapter 6, ‘SimulationWare – An Abstract Machine for Symbolic,’ presents the machine construction, while Chapter 7 develops its connection with KnowledgeWare. The same approach was subsequently documented in Symbolic Analysis as a Basis for Program Comprehension [6,12].

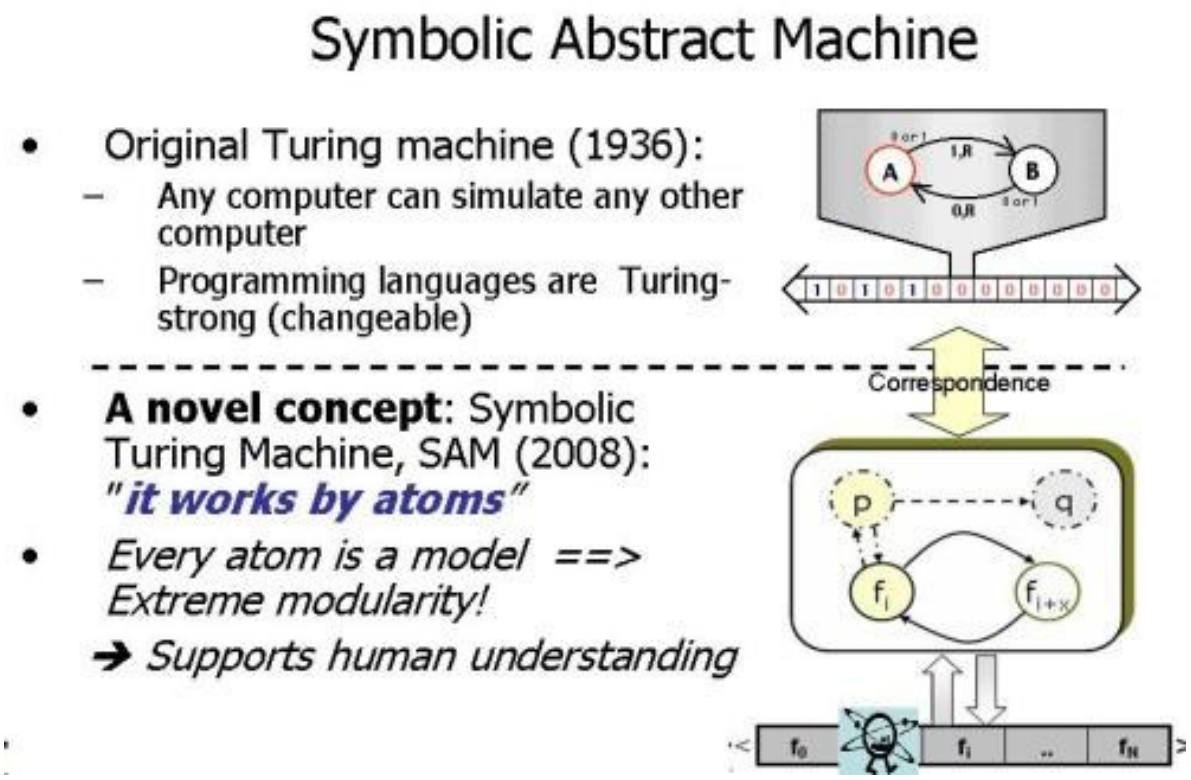

Figure 4. Universal Turing Machine and Symbolic Abstract Machine. The upper part represents universal rule-governed computation. The lower part extends simulation to typed semantic atoms whose local evaluation and relations support program comprehension and systemic interpretation.

SAM operates through symbolic atoms. Each atom is a typed and independently interpretable model element with locally defined semantics, while relations between atoms reconstruct the behaviour of the larger program or system. Its elementary operation is symbolic evaluation: terms, bindings, rules and states are evaluated according to the semantics of the represented language. The resulting atomism does not imply that the whole is reduced to isolated fragments. Each atom retains its role, references and position in the larger model, thereby combining modularity with program comprehension and systemic interpretation.

This construction changes the implementation philosophy of computation. Computation is no longer limited to numerical calculation or manipulation of uninterpreted machine symbols. It encompasses symbolic representation, typing, substitution, variable binding, evaluation, inference, simulation and model transformation. A symbolic machine can therefore operate on descriptions of programs, models and systems while retaining a traceable relation between the symbolic operation and its interpreted meaning.

Metamodeling makes this higher symbolic level explicit. A metamodel describes the constructs, rules and constraints by which models are formed. Its descriptors connect model elements with their objects, interpretations and permissible operations [15]. Geographic information provides a familiar example: a map object, its descriptor and its displayed occurrence are distinct but connected symbolic entities. Their meaning arises from the organized relations of the map system rather than from any isolated graphic mark. Model-driven engineering applies the same principle to software by making models, metamodels and transformations operational parts of computation.

Andrew Wells' Rethinking Cognitive Computation returns to Turing's original analysis and examines computation in relation to mind, environment and social interaction [14]. His ecological functionalism extends the discussion beyond an isolated internal mechanism toward distributed and concurrent processes. This supports a systemic interpretation of computation in which symbols are evaluated within an environment and acquire functional significance through their relations and use.

GoodReason continues the development from the Symbolic language and SAM toward a general semantic language and coordinate system. A GoodReason symbol such as π4 acts as an addressable anchor that connects a theoretical perspective, a cognitive depth, a System of Interest and its contextual interpretations. A geographical landmark, an organization, a conflict or a scientific theory can be designated as the SOI, while its occurrences describe different situated manifestations and interpretations. The symbolic description remains extensible: new descriptors, relations and viewpoints can be introduced while each actual execution remains governed by explicit types, rules, evidence and available computational resources.

When such a symbolic structure is supplied with purpose, rules, state, action capabilities and feedback, it can also serve as the semantic model of an artificial agent. The α–Ω coordinate system provides a concise means for communicating purposes, theories, environmental information, transformation pressures, structures, designs, implementations and feedback. It allows human intentions and interpretations to remain visible within computational activity and enables the results to be examined, questioned and revised.

The figure contrasts two complementary aspects of computation. Its upper part emphasizes the mathematical definition and universality of effective computation. Its lower part emphasizes logic, semantics, symbolic evaluation and systemic interpretation. SAM is thus an implemented bridge from abstract computation to symbolically interpretable program behaviour, while GoodReason extends the same philosophy toward systems, communication and human-guided artificial intelligence.

Statistical language models follow a different computational architecture and can participate in this symbolic environment through explicitly configured interfaces. Their implications for intelligence, consciousness and higher-level agent cognition are reserved for a subsequent paper. The present construction establishes the operational foundation: symbolic computation is already a working paradigm in which human beings can define meanings, direct evaluation and retain authority over the purposes for which computational systems are used.

9. Relationship to System Language

Mobus and Anderson's System Language proposes a lexicon, syntax and semantics for describing system components, flows, boundaries, interfaces and behavior. It aims to support analysis, formal modelling and eventual compilation into executable simulation code [10]. This is a close and valuable neighboring program.

System Language and SCS address overlapping but distinct parts of executable systems modelling. System Language primarily develops a general descriptive and simulation-oriented language of systemness. SCS places logic-programming states and epistemic outcomes at the center: free and bound variables, unification, nondeterministic solutions, open queries, supported conflicts and first-class justifications.

Dimension	System Language	Symbolically computable system
Primary emphasis	General language of system structure and dynamics	Operational logic of partial systemic knowledge
Core objects	Components, flows, boundaries, interfaces, processes	Claims, variables, relations, questions, bindings and reasons
Execution path	Formal model compiled toward simulation	Inference and state transition, possibly connected to simulation or action

Dimension	System Language	Symbolically computable system
Epistemic openness	Model-dependent and interpretive	Explicit statuses for open, refuted and conflicting results
Potential relationship	Supplies systemic lexicon and structural semantics	Supplies reasoning and execution semantics over such representations

Possible synthesis: System Language can describe what a system is composed of and how it behaves; an SCS layer can state what is known, what is asked, how alternatives are derived and why a transition is accepted.

10. Limits, grounds of validity and extension

10.1 Conceptual limits

- Symbolically computable applies to the represented system model; claims about the wider real system require empirical and interpretive grounds.
- A symbol assignment is a semantic convention. For example, φ is an address for design and solution; its geometric association does not prove a design correct.
- An unresolved query records the inquiry state and its missing conditions; alternatives still require evidence and admissibility criteria.
- Closed-world failure and open-world falsity are separate statuses and must be declared as such.
- Conflicting records become knowledge only through an explicit argumentation or inconsistency policy.
- Decidability does not imply practical tractability. NP-complete search, path explosion and unbounded interaction can make a formally specified task infeasible at the required scale.
- A proof is only as comprehensive as its specification: omitted context and side effects remain outside the established conclusion.

10.2 Grounds of validity and operational evidence

The construction is grounded by historical continuity, conceptual coherence and constructive realisation. Turing and Church establish the formal boundary; the Chomsky hierarchy supplies the generative-language relation; logic programming supplies free and bound variables, unification and nondeterminism; SAM supplies an implemented symbolic machine; and GoodReason extends the same line into a

systemic semantic coordinate space. Operational validity is examined through the following capabilities:

- Type-correct representation of a bounded SOI and its α-Ω coordinates.
- Queries containing free variables and repeated production of alternative bindings.
- Separation of answer, refutation, open status and supported conflict.
- Traceable justification from each result to rules, evidence and responsible agents.
- Controlled handling of recursive or circular constraints, including non-termination detection.
- Declared handling of side effects at module, service and agent boundaries; intended input-output relations remain part of the module logic.
- Round-trip transformation between machine-readable structures and intelligible human views.
- Reproducible multi-agent exchange in which another agent can inspect or challenge a result.
- A complexity profile recording search growth, proof or certificate checking cost, stopping policy and resource budget.
- Metacognitive inspection in which a trace or justification can trigger revision, an evidence request or an explicit open status.

10.3 Current realisation and extension path

The work already spans the formal core, symbolic analysis, the implemented SAM construction, the GoodReason α–Ω coordinate system, recursive agent profiles and typed logic-programming practice. These are mutually supporting realisations of one paradigm rather than a sequence waiting for an external institution to authorize its beginning.

Extension now means making this line easier to inspect, reuse and challenge: publishing worked cases, machine-readable structures, queries, expected result states, justifications and complexity profiles; connecting symbolic models with human and artificial agents; and comparing the resulting practice with neighboring systems languages. Independent review strengthens the work without defining whether the construction exists.

Example. We recognize Luhmann’s pioneering work in modelling society and the economic system. Technological development now enables his theories, and those of many other thinkers, to be carried into computer-supported practice. In this integration, GoodReason can serve as an intermediate language and semantic structure for defining and connecting the key computations of the System of Interest.

Ground	Present basis	Extension
Formal	Result relation, free/bound states, proof conditions and complexity boundaries	Reference semantics and reproducible query suites
Constructive	SAM, symbolic atoms, evaluation and program comprehension	Documented implementations and comparative cases
Systemic	GoodReason α–Ω coordinates and seven depths of inquiry	Worked system models with inspectable assumptions
Operational	Reasoning, state transition, feedback and current agent use	Traceable human-agent workflows and review
Scientific	Published dissertation, book and systems-science papers	Critical comparison, replication and cumulative refinement

11. Conclusion

A symbolically computable system is a bounded, typed and executable knowledge model in which symbols can be represented, analysed, reasoned over and used to transform states. Its distinctive feature is the preservation of epistemic openness: a free variable can remain free, several bindings can coexist, a contradiction can remain visible and a question can be stored as the best available knowledge when its missing conditions are explicit.

This proposal reconnects systems science with a strand of symbolic AI that began with formal computability, generative grammar, machine-oriented inference and logic programming. GoodReason adds a geometric and systemic address space that extends from α-purpose to Ω-feedback and from operational stability to philosophical reflection. Its open architecture lets human interpretation, formal reasoning, symbolic evaluation and agent-based execution correct and strengthen one another. Justification traces keep the conditions and limits of earlier reasoning available to both people and machines.

A system becomes symbolically computable when its representation preserves the meaning required for analysis, the structure required for reasoning, the operational semantics required for execution and the provenance required for criticism and revision. Computation is the active relation joining those symbolic and systemic dimensions.

References

[1] Turing, A. M. (1936/1937). On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 42, 230-265. https://doi.org/10.1112/plms/s2-42.1.230

[2] Newell, A., & Simon, H. A. (1976). Computer Science as Empirical Inquiry: Symbols and Search. Communications of the ACM, 19(3), 113-126. https://doi.org/10.1145/360018.360022

[3] Robinson, J. A. (1965). A Machine-Oriented Logic Based on the Resolution Principle. Journal of the ACM, 12(1), 23-41. https://doi.org/10.1145/321250.321253

[4] Kowalski, R. (1979). Algorithm = Logic + Control. Communications of the ACM, 22(7), 424-436. https://doi.org/10.1145/359131.359136

[5] Prolog Development Center. Visual Prolog Language Reference: terms, flow patterns, unification and free/bound variables. https://wiki.visual-prolog.com/index.php?title=Language_Reference_Book

[6] Laitila, E. (2008). Symbolic Analysis and Atomistic Model as a Basis for a Program Comprehension Methodology. Jyväskylä Studies in Computing 90, University of Jyväskylä. ISBN 978-951-39-3252-7.

[7] King, J. C. (1976). Symbolic Execution and Program Testing. Communications of the ACM, 19(7), 385-394. https://doi.org/10.1145/360248.360252

[8] Shoham, Y. (1993). Agent-Oriented Programming. Artificial Intelligence, 60(1), 51-92.

[9] Dung, P. M. (1995). On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence, 77(2), 321-357.

[10] Mobus, G., & Anderson, K. (2016). System Language: Understanding Systems. Proceedings of the 60th Annual Meeting of the ISSS. https://journals.isss.org/index.php/proceedings60th/article/view/2917

[11] Laitila, E. (2026). Axiomatic Systems Science - Geometry of Thinking. Journal of the International Society for the Systems Sciences. https://journals.isss.org/index.php/jisss/article/view/4627

[12] Laitila, E. (2009). Symbolic Analysis as a Basis for Program Comprehension. VDM Verlag. ISBN 978-3-639-16833-4.

[13] Gudwin, R. R., & Queiroz, J. (2005). Towards an Introduction to Computational Semiotics. Proceedings of the IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems. https://doi.org/10.1109/KIMAS.2005.1427113

[14] Wells, A. J. (2005). Rethinking Cognitive Computation: Turing and the Science of the Mind. Palgrave Macmillan.

[15] Gonzalez-Perez, C., & Henderson-Sellers, B. (2008). Metamodelling for Software Engineering. Wiley.

[16] Cook, S. A. (1971). The Complexity of Theorem-Proving Procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing, 151-158. https://doi.org/10.1145/800157.805047

[17] Floyd, R. W. (1967). Assigning Meanings to Programs. Proceedings of Symposia in Applied Mathematics, 19, 19-32.

[18] Hoare, C. A. R. (1969). An Axiomatic Basis for Computer Programming. Communications of the ACM, 12(10), 576-580. https://doi.org/10.1145/363235.363259

[19] O'Hearn, P. W., Reynolds, J. C., & Yang, H. (2001). Local Reasoning about Programs that Alter Data Structures. Computer Science Logic, 1-19. https://doi.org/10.1007/3-540-44802-0_1

[20] Atkin, A. (2022). Peirce's Theory of Signs. Stanford Encyclopedia of Philosophy. https://plato.stanford.edu/entries/peirce-semiotics/

[21] Church, A. (1936). An Unsolvable Problem of Elementary Number Theory. American Journal of Mathematics, 58(2), 345-363. https://doi.org/10.2307/2371045

[22] Capra, F., & Luisi, P. L. (2014). The Systems View of Life: A Unifying Vision. Cambridge University Press.

[23] Stanford University. Symbolic Systems Program: interdisciplinary study of natural and artificial systems that represent, process and act on information. https://symsys.stanford.edu/

[24] Flavell, J. H. (1979). Metacognition and Cognitive Monitoring: A New Area of Cognitive-Developmental Inquiry. American Psychologist, 34(10), 906-911. https://doi.org/10.1037/0003-066X.34.10.906

[25] Cox, M. T., Mohammad, Z., Kondrakunta, S., Gogineni, V. R., Dannenhauer, D., & Larue, O. (2022). Computational Metacognition. arXiv:2201.12885. https://doi.org/10.48550/arXiv.2201.12885

[26] Belnap, N. D. (1977). A Useful Four-Valued Logic. In J. M. Dunn & G. Epstein (Eds.), Modern Uses of Multiple-Valued Logic, 8-37. Springer. https://doi.org/10.1007/978-94-010-1161-7_2

[27] Kakas, A. C., Kowalski, R. A., & Toni, F. (1992). Abductive Logic Programming. Journal of Logic and Computation, 2(6), 719-770. https://doi.org/10.1093/logcom/2.6.719

[28] Chomsky, N. (1959). On Certain Formal Properties of Grammars. Information and Control, 2(2), 137-167. https://doi.org/10.1016/S0019-9958(59)90362-6

[29] The Stanford Encyclopedia of Philosophy. The Computational Theory of Mind (*Dec 18, 2024*). https://plato.stanford.edu/archives/fall2025/entries/computational-mind

Status of this document

Version 1.0 is a concept paper for the symbolic paradigm of systems computation. It consolidates the Turing–Church foundation, the Chomsky hierarchy, computational semiotics, symbolic analysis, the implemented Symbolic Abstract Machine, partial-knowledge semantics and the GoodReason coordinate space into one coherent construction. Intelligence, consciousness and higher-level agent cognition are reserved for a companion paper so that this paper remains centered on computation.