Formal Identity, Discrete Stabilization, and Consensus Risk in Verified Distributed Kernels

Veaceslav Molodiuc

Independent Researcher, LoculoSoft Initiative
Chisinau, Republic of Moldova
veaceslav.molodiuc@gmail.com

AI-assisted preprint, Version 1.1
July 27, 2026

Independent preprint prepared in the author's own layout.

Abstract

Verified distributed kernels require more than authenticated messages: they require formal identity, recoverable operational states, and measurable consensus risk. This paper connects formal identity, discrete stabilization, the LoculoSoft kernel vocabulary, and PRISM probabilistic model checking. LoculoSoft is treated cautiously as a proposed structural framework and research direction. A reproducible PRISM 4.10.1 check on a three-process Herman-style token ring shows stable reachability with probability 1 and maximum expected stabilization time near 4/3 steps.

Keywords: formal identity, discrete stabilization, consensus risk, distributed kernels, PRISM, LoculoSoft.

1 Introduction

Distributed intelligent systems increasingly combine autonomous software agents, shared state, cryptographic authentication, smart services, and asynchronous communication. Their central engineering difficulty is not only to compute, but to preserve who is computing, which state is legal, and under what conditions several agents may safely agree. A system may have strong cryptography and still lose its operational identity if local memory, transient transformation, and authorization are mixed in the same uncontrolled execution space. Conversely, a system may have a formally defined protocol and still fail if a temporary fault prevents the network from returning to a legal state.

This paper presents a system-analysis perspective on the LoculoSoft framework and PRISM probabilistic verification. The central line is

$$\text{formal identity} \to \text{discrete stabilization} \to \text{consensus risk} \to \text{LoculoSoft kernel} \to \text{PRISM verification}.$$

The aim is deliberately modest. We do not claim that the LoculoSoft framework is a completed or fully proved theory. We treat it as a proposed structural registry, a working vocabulary, and a research direction for connecting identity preservation, local stabilization, and model-checkable properties of distributed protocols.

The technical anchor is PRISM, a probabilistic model checker for discrete-time Markov chains, Markov decision processes, and related stochastic models [6]. In Section 6 we report a local PRISM 4.10.1 check, run after resolving the Windows `javaw.exe` startup problem by installing Eclipse Temurin JDK. The example is intentionally small: a three-process randomized token ring. Its value is not that it validates LoculoSoft as a whole, but that it demonstrates how a structural notion of stabilization can be connected to reproducible probabilistic verification.

2 Formal Identity of Distributed Agents

Definition 1. *An agent is an operational component able to observe a local environment, update an internal state, and participate in a protocol shared with other components. An agent may be a process, a network node, a validator, a kernel module, or an abstract logical participant.*

Definition 2. *A formal identity is a verifiable invariant that distinguishes an agent from its environment and from other agents across admissible state transitions. It may use cryptographic credentials, but it is not reducible to a public key or a registry entry.*

Cryptographic protocols provide important infrastructure. For example, the SIGMA family of authenticated Diffie–Hellman protocols supports identity protection and authenticated key exchange in adversarial settings [5]. However, authentication only answers part of the question. It can show that a message is associated with a credential; it does not by itself prove that a process preserves its operational role, its allowed memory region, or the invariant that separates persistent identity from transient computation.

For verified kernels, identity must therefore be treated as a structural property. Homotopy Type Theory (HoTT) suggests one useful conceptual direction: equality and identity can be understood through paths and equivalences rather than only as Boolean comparison [4]. Proof assistants such as Coq/Rocq provide a practical setting in which such concepts may be encoded as types, terms, and proof obligations [12]. The claim is not that HoTT automatically solves kernel identity, but that type-theoretic verification makes it possible to ask sharper questions: which transitions preserve identity, which transitions rewrite it, and which transitions are disallowed by construction?

Within the proposed LoculoSoft notation we distinguish between conjunctive and disjunctive aspects of identity:

$$\mathbf{I}_{\wedge}, \qquad \mathbf{I}_{\vee}, \qquad \mathbf{T} \cap \mathbf{I}_{\vee} = \varnothing.$$

Here $\mathbf{T}$ denotes the region of transient transformations. The separation condition is not presented as a standard theorem. It is a design convention: temporary transformations should not occupy the same semantic space as the formal identity they manipulate. In a verified implementation, this convention would have to be translated into types, memory rules, or transition invariants.

The second notation expresses a related requirement:

$$\ulcorner F|_K \urcorner \in K.$$

It says, cautiously, that the restriction of an operation F to a kernel K should be represented inside the same kernel vocabulary. In software terms, the local law of a module should be inspectable by the module's verification environment rather than hidden in informal execution behavior.

3 Discrete Stabilization

Definition 3. *An operational state is a finite description of the local variables, registers, relations, and protocol indicators that determine the next admissible transition of an agent or a group of agents.*

Definition 4. *A fault is a transient or persistent violation of the expected execution assumptions: a corrupted bit, a lost message, an inconsistent local state, an adversarial update, or false external data.*

Definition 5. *A stable state is a legal configuration in which the target protocol property holds and remains preserved under normal transitions until a new fault appears.*

Self-stabilization, introduced by Dijkstra, is the property that a distributed system starting from an arbitrary configuration can reach a legitimate configuration without external intervention [2]. Herman's probabilistic self-stabilization result later showed that randomization can be used to recover stability in token-ring protocols [3]. These ideas are important for verified distributed kernels because they shift the question from "does the program pass tests?" to "from which states can the system recover, with what probability, and in what expected time?"

Discrete stabilization is naturally expressed over countable configurations. A local kernel transition changes a finite state. A distributed protocol combines several such transitions under asynchronous or randomized scheduling. If a fault moves the system outside the legal region, a stabilizing protocol must return it to legality. If the return is guaranteed in a finite number of steps, we speak about finite-time stabilization. If the guarantee is probabilistic, the statement must specify reachability probability and expected stabilization time.

This matters for consensus. A consensus protocol is not only a decision procedure; it is also a stabilization mechanism around a shared value or relation. If agents cannot preserve local identity, then the participants of consensus become ambiguous. If operational state is not defined, then agreement may be confused with inactivity. If the stable region is not explicit, then recovery from a fault cannot be measured.

4 Consensus Risk

Definition 6. *Consensus risk is the possibility that a distributed system reaches an incorrect, unstable, manipulated, or non-finalizable common decision, even though its local components appear operational.*

Consensus risk appears in many settings, from multi-agent control to blockchains and replicated services. The classical Byzantine generals problem already shows that agreement can fail when some participants behave inconsistently or maliciously [7]. In networked multi-agent systems, consensus and cooperation depend on topology, communication rules, and stability assumptions [8]. A kernel-oriented analysis must connect these external protocol conditions with local state preservation.

Several examples illustrate the risk surface. Validator collusion occurs when nodes that are expected to be independent coordinate decisions. Sybil risk appears when one adversary creates many apparent identities and changes the effective voting weight. Oracle risk appears when a formally correct protocol consumes false external data. Client-diversity risk is subtler: several independent client implementations may protect the network against a single code defect, while also increasing the total attack surface. Bridge risk appears when two ledgers, runtimes, or semantic domains are connected by translation logic that is difficult to verify end to end.

These examples should not make the paper a blockchain paper. Their value is diagnostic. They show that consensus risk sits at the boundary between identity, state, and transformation.

If the identity of a participant is unstable, the decision set is unstable. If the state space is not separated into legal and illegal regions, recovery is unmeasured. If transformations between domains are not represented in a formal vocabulary, a system can execute a technically valid but semantically wrong transition.

5 The LoculoSoft Kernel as a Working Framework

LoculoSoft is used here as a proposed architectural framework for reasoning about a kernel of agentic identity. The terminology is intentionally cautious: proposed framework, structural registry, working vocabulary, and research direction. It is not treated as a closed theory, and no theorem in this paper depends on accepting the full LoculoSoft vocabulary as standard mathematics.

The broader LoculoSoft program was initially formulated in the author's 2025 master's thesis [11]. The present paper uses the term in a narrower operational sense, focused on identity preservation, discrete stabilization, and consensus analysis in distributed kernels.

Definition 7. *A stabilizer L_k, in the proposed LoculoSoft vocabulary, is a structural or operational threshold associated with a separation, memory, orientation, or nodal-stabilization property. The index k is an internal label of the proposed registry, not a standard invariant of distributed computing.*

The central design idea is separation. A verified kernel should distinguish persistent identity, current transformation, and operational state. In this vocabulary, L_{36} is used as an operational stabilizer that separates transformation from identity:

$$L_{36} = L_2^2 \times L_3^2 = 2^2 \cdot 3^2 = 36.$$

This equation is best read as notation inside the framework. It does not by itself prove memory safety, consensus correctness, or self-stabilization. Its role is to mark a proposed structural threshold that must later be connected to formal transition systems, proof obligations, or experiments.

The current registry can be summarized as follows. L_{18} is a local semi-register, associated with minimal local orientation. L_{36} is an operational stabilizer, associated with separating transformation from identity. L_{275} is a planar or global nodal threshold inspired by four-page book embedding results for planar graphs [1]; the connection is an analogy for structural capacity, not a direct theorem about kernels. L_{548} is described as polar doubling, and L_{584} as a proposed general planar stabilizer combining local and global stabilization roles.

This registry has value if it disciplines future work. Each L_k should be tied to a finite model, a formal invariant, a proof attempt, or a reproducible experiment. Without such ties, the numbers remain suggestive terminology. With such ties, they may become a useful research vocabulary for mapping local identity preservation to global stabilization.

6 A Reproducible PRISM Check

The PRISM example was run locally in PRISM 4.10.1 after the Windows startup problem was fixed by installing Eclipse Temurin JDK, giving PRISM access to `javaw.exe`. The model file is `examples/herman3_all.pm`, and the property file is `examples/herman3_all.props`. It is a deliberately minimal discrete-time Markov chain (DTMC) with three processes and all initial states enabled.

A compact excerpt of the model is:

```
dtmc
```

```
const double p = 0.5;

module process1
    x1 : [0..1];
    [step] (x1=x3)  -> p : (x1'=0) + (1-p) : (x1'=1);
    [step] !(x1=x3) -> (x1'=x3);
endmodule

module process2 = process1 [ x1=x2, x3=x1 ] endmodule
module process3 = process1 [ x1=x3, x3=x2 ] endmodule
```

The full model defines the token count and labels a configuration as stable when exactly one token is present. The verified properties were:

```
filter(min, P=? [ F "stable" ])
filter(max, R{"steps"}=? [ F "stable" ])
```

PRISM built a DTMC with 8 states, 8 initial states, and 28 transitions. The first property returned

```
filter(min, P=?  [ F "stable" ]) = 1.0,
```

which means that from every initial state the stable configuration is reached with probability 1. The second property returned

```
filter(max, R{"steps"}=?  [ F "stable" ])
          = 1.3333332755111582,
```

so the maximum expected stabilization time over all initial states is approximately 4/3 steps.

This result does not validate the full LoculoSoft framework. It verifies only a small DTMC with three processes. It does not cover Byzantine behavior, realistic kernels, the full stabilizer registry, or type-theoretic proof obligations. Its contribution is methodological: it shows how a proposed structural notion of stabilization can be reduced to a finite stochastic model, checked by PRISM, and reported with reproducible numerical results.

7 Limitations

The first limitation is epistemic. LoculoSoft is treated as a working framework, not as established standard theory. The stabilizer labels L_{18}, L_{36}, L_{275}, L_{548}, and L_{584} require formal definitions, independent proofs, and experimental validation before they can be used as mathematical claims.

The second limitation is scale. PRISM is powerful for finite probabilistic models, but model checking is constrained by state-space explosion. A three-process DTMC is useful as a minimal reproducible check; it is not evidence that large distributed kernels will stabilize under the same bounds. Larger models may require abstraction, symmetry reduction, statistical model checking, or proof-assistant support.

The third limitation is semantic. Formal verification of a local transition system does not guarantee that external data, cross-domain translation, or oracle input is true. A verified kernel can still execute a semantically bad transition if the environment supplies false assumptions. This is why consensus risk must be studied at several levels: identity, state, communication, semantics, and external data.

The fourth limitation concerns bibliography and terminology. The paper relies on standard references for self-stabilization, PRISM, consensus, HoTT, Coq/Rocq, SIGMA, Byzantine agreement, and planar graph book embeddings. Any additional LoculoSoft-specific publication should be added only after its bibliographic data and claims are independently checked.

8 Conclusion

The paper proposed a compact system-analysis framework for verified distributed kernels. Its main contribution is the connection between formal identity, discrete stabilization, consensus risk, a cautious LoculoSoft kernel vocabulary, and a reproducible PRISM experiment. Formal identity clarifies who or what participates in a protocol. Discrete stabilization clarifies how illegal states can return to legal configurations. Consensus risk clarifies how local correctness can still fail at the level of shared decision. LoculoSoft offers a proposed structural registry for separating identity, transformation, and stabilization. PRISM supplies a practical bridge from vocabulary to finite probabilistic verification.

The local PRISM 4.10.1 check is intentionally small, but it is concrete. It confirms that the three-process demonstrative model reaches a stable configuration with probability 1 from every initial state and has maximum expected stabilization time close to 4/3 steps. The next step is not to overstate this result, but to expand the same discipline: define each proposed stabilizer precisely, encode finite models, prove local invariants where possible, and test larger distributed scenarios.

AI Assistance and Author Responsibility

This manuscript was developed with substantial assistance from OpenAI's ChatGPT in conceptual organization, linguistic refinement, consistency checking, and preparation of the English text. The human author reviewed the manuscript and assumes full responsibility for all mathematical claims, calculations, interpretations, references, and conclusions.

Reproducibility Statement

The complete PRISM model and the two verified properties used in Section 6 are reproduced in Appendix A. The numerical results reported in the paper were obtained locally with PRISM 4.10.1. The appendix makes the verification specification available inside this PDF without requiring embedded or executable files.

References

[1] M. A. Bekos, M. Kaufmann, F. Klute, S. Pupyrev, C. Raftopoulou, and T. Ueckerdt, "Four pages are indeed necessary for planar graphs," *Journal of Computational Geometry*, vol. 11, no. 1, pp. 332–353, 2020. DOI: https://doi.org/10.20382/jocg.v11i1a12.

[2] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, vol. 17, no. 11, pp. 643–644, 1974. DOI: https://doi.org/10.1145/361179.361202.

[3] T. Herman, "Probabilistic self-stabilization," *Information Processing Letters*, vol. 35, no. 2, pp. 63–67, 1990. DOI: https://doi.org/10.1016/0020-0190(90)90107-9.

[4] The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. [Online]. Available: https://homotopytypetheory.org/book/.

[5] H. Krawczyk, "SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie–Hellman and its use in the IKE protocols," in *Advances in Cryptology – CRYPTO 2003*, LNCS 2729, Springer, 2003, pp. 400–425. DOI: https://doi.org/10.1007/978-3-540-45146-4_24.

[6] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*, LNCS 6806, Springer, 2011, pp. 585–591. DOI: https://doi.org/10.1007/978-3-642-22110-1_47.

[7] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982. DOI: https://doi.org/10.1145/357172.357176.

[8] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007. DOI: https://doi.org/10.1109/JPROC.2006.887293.

[9] PRISM Model Checker Team, "Randomised self-stabilising algorithms." [Online]. Available: https://www.prismmodelchecker.org/casestudies/self-stabilisation.php. Accessed: June 15, 2026.

[10] PRISM Model Checker Team, "Randomised consensus shared coin protocol." [Online]. Available: https://www.prismmodelchecker.org/casestudies/consensus-prism. Accessed: June 15, 2026.

[11] V. Molodiuc, *Probleme speciale în teoria numerelor și a structurilor algebrice* [Special Problems in Number Theory and Algebraic Structures], unpublished master's thesis, "Ion Creangă" State Pedagogical University of Chișinău, Faculty of Physics, Mathematics and Information Technologies, Chișinău, Republic of Moldova, 2025. [In Romanian].

[12] The Rocq Prover Development Team, "The Rocq prover reference manual." [Online]. Available: https://rocq-prover.org/doc/master/refman/. Accessed: June 15, 2026.

A Complete PRISM Model and Properties

The following model is the complete three-process DTMC used for the reproducible check.

Listing 1: PRISM model `herman3_all.pm`

```
dtmc

const double p = 0.5;

module process1
    x1 : [0..1];

    [step] (x1=x3)  -> p : (x1'=0) + (1-p) : (x1'=1);
    [step] !(x1=x3) -> (x1'=x3);
endmodule

module process2 = process1 [ x1=x2, x3=x1 ] endmodule
module process3 = process1 [ x1=x3, x3=x2 ] endmodule

init
    true
endinit

formula num_tokens =
    (x1=x3 ? 1 : 0) +
    (x2=x1 ? 1 : 0) +
    (x3=x2 ? 1 : 0);
```

```
label "stable" = num_tokens=1;

rewards "steps"
    true : 1;
endrewards
```

The verified properties are:

Listing 2: PRISM properties `herman3_all.props`

```
filter(min, P=? [ F "stable" ])

filter(max, R{"steps"}=? [ F "stable" ])
```

B Verification Record

The expected model statistics are: model type DTMC, 8 states, 8 initial states, and 28 transitions. The expected results are

$$\text{filter}(\min, \Pr[\mathbf{F}\,\text{stable}]) = 1.0$$

and

$$\text{filter}(\max, \mathbb{E}[\text{steps until stable}]) = 1.3333332755111582 \approx \frac{4}{3}.$$

These values concern only the demonstrative three-process model described in Section 6.