Dimensional Game Theory Application of Non-Deterministic Finite Automaton Directed Acyclic Graph for Actor Modelling via Phenomenological Observer and Consumer HDIS Functor Coherence Operator Interpolation with Fault-Tolerant Resolution Implementation

OBINexus Research Group
`github.com/obinexus/dgt`

October 2025

Abstract

This document formalises the Dimensional Game Theory (DGT) framework and its Non-Deterministic Finite Automaton (NFA) functor extension, integrates Directed Acyclic Graph (DAG) topology for HDIS (Hybrid Directed Instruction Systems), and provides algorithmic and verification tooling (ODTS, CRIF). It is intended as a single-file LaTeX source suitable for Overleaf compilation and inclusion in the project repository. The text contains formal definitions, algorithms, pseudocode, diagrams, and a practical integrity metric (CRIF) for witness ledger verification.

Contents

1 Introduction

1.1 Background: From Classical Game Theory to Dimensional Extensions

A short motivation for DGT: traditional game theory analyses strategic agents with fixed action sets and payoffs. DGT extends this by adding "strategic dimensions" that are activated contextually and by building functorial mappings between observer and actor spaces.

1.2 Role of Phenomenology in Computational Modelling

Phenomenological elements (phenotype, phenomemory, phenovalue) are included as first-class types to model human-in-the-loop systems and to ground coherence metrics in measurable observables.

1.3 HDIS as Foundation: Human-in-the-Loop Coherence Systems

HDIS is introduced as the architectural seed for active systems that observe, adapt, and self-heal. Implementations include DIRAM, DIPAD and the SemVerX hot-swap engine.

1.4 Research Aim and Scope

This document unifies formalism, algorithms and reference implementations (links to repositories) for DGT, NFA functor extension, ODTS and HDIS integration.

2 Theoretical Framework

2.1 Dimensional Game Theory (DGT): Core Definitions

Glossary and Notation

$\mathbb{N}$ Natural numbers $\{0, 1, 2, \ldots\}$.

$\mathbb{Z}$ Integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

$\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ standard sets of rational, real and complex numbers.

DGT-specific symbols

F, G Functorial mappings between behaviour spaces.

$F \cdot G$ Composition: $(F \cdot G)(x) = F(G(x))$; in DGT composition may be non-deterministic through functor-lifting to power-set codomains.

$F \star G$ Coherence operator (lossless interaction), defined in Section 2.6.

Φ, Ψ Phenomenological functors: Φ maps raw observations to phenovalue; Ψ maps intent/actor transforms.

2.2 Non-Deterministic Finite Automata (NFA) as Behavioural Models

Define an actor as $\mathcal{A}_{\text{NFA}} = (Q, \Sigma, \delta, q_0, F)$ where $\delta : Q \times \Sigma \to 2^Q$. Actor policies are modelled as NFA transitions and functor lifts let us compose NFAs with DAG topologies.

2.3 Directed Acyclic Graphs (DAG) in System Coherence

Nodes model actor components; edges model observer-consumer transactions and coherence receipts. A DAG $G = (V, E)$ is used so that topological traversal and functor composition are well-defined.

2.4 Actor-Observer-Consumer Paradigm

Each node implements an *Observer* that produces phenovalue and a *Consumer* that acts on it. The pair yields dual receipts stored in the witness ledger.

2.5 Phenomenological Type System (Phenotype, Phenomemory, Phenovalue)

Formal type definitions and interfaces for phenomodels, including minimal API examples (pseudocode) for observing and consuming layers.

2.6 Coherence Operators and Functor Composition

Define algebraic operators used to combine functors: union $\mathcal{U}$, coherence $\star$, disjoint $\div$, and null $\perp$. A working definition:

$$H = F \star G = \mathcal{C}(F, G) \cdot (F \cdot G)$$

with $\mathcal{C}(F, G) \in [0, 1]$ the coherence measure (see Section 6).

3 Mathematical and Algorithmic Foundation

3.1 State Transition Functions and NFA Functor Definition

Formalisation of lifted transitions: if δ is the NFA transition, we define the functor-lifted transition $\delta^\sharp$ mapping distributions or power-set states.

3.2 Functor Composition: $F \cdot G$ Operator Interpolation

Describe the interpolation semantics when F and G are stochastic/non- deterministic functors. Provide pseudocode for composition and for estimating $\mathcal{C}(F, G)$ via sampled traces.

3.3 Set-Theoretic Mapping: Union, Disjoint, Pairing, Division

Formal set operations used to reason about lossless vs lossy joins in the DAG.

3.4 Complexity Constraints and $O(\log n)$ Auxiliary Space

State the Functor Framework's constraint: implement traversals and QA aggregations using $O(\log n)$ auxiliary space (practical implementation notes for iterative topological traversal with bounded frontier are provided).

3.5 Lossless vs Lossy Systems (Huffman–AVL Equilibrium)

Explain the equilibrium idea: Huffman-like compression (entropy minimisation) balanced with AVL-like structural constraints to preserve recoverability.

3.6 Coherence Preservation under Non-Determinism

Lossless and lossy examples and the probing functor $P(\mathcal{A}_{\mathrm{NFA}}, \theta)$ that measures sensitivity $\nabla_\theta \mathcal{C}$.

4 OBINexus Derivative Tracing System (ODTS)

A safety-first subsystem to trace symbolic derivatives and confirm termination for polynomial-type models used by controllers. Example use-case and API.

5 HDIS Integration

5.1 HDIS as a Consumer-Observer Digital Nervous System

Architectural overview, the witness ledger concept and coherence receipts.

5.2 Mapping Observer and Consumer Roles

Concrete mapping table and role responsibilities.

5.3 SemVerX: Evolutionary Component Replacement

Specification of SemVerX, hot-swap policies and DAG-based resolver.

5.4 Witness Ledger and Coherence Receipts

Full CRIF specification and pseudocode (see Section 6).

5.5 Polyglot Implementation: Rust – ESP32 – Python

Integration notes, canonical IR, and polyglot binding rules for the HDIS stack.

6 Coherence Receipt Integrity Function (CRIF)

6.1 Notation

$G = (V, E)$ DAG, witness ledger $L = \{r_e\}$, receipts $r_e = (o, c, \phi, \tau, \sigma)$.

6.2 CRIF Definition

$$C = \lambda\, I_{\text{hash}}(L) + (1-\lambda)\, Q_{\text{agg}}$$

with detailed computation steps and pseudocode implemented as Algorithm 1.

Algorithm 1 ComputeCRIF

```
1: procedure ComputeCRIF($G, L, Pipelines, w, \alpha, \lambda, C_{req}$)
2:     compute $I_{hash}$ by validating hashes for receipts in $L$
3:     compute per-pipeline $Q(P)$ using F1 gate metrics
4:     compute $Q_{agg} = \sum_i \alpha_i Q(P_i)$
5:     $C = \lambda I_{hash} + (1-\lambda) Q_{agg}$
6:     return $C$ and pass/fail flag $C \geq C_{req}$
7: end procedure
```

7 Actor Behaviour Modelling

7.1 Non-Deterministic Action-State Space

Formal NFA-based actor models and compositional rules.

7.2 Dimensional Extension of Actor Roles

How dimensions activate and interact; detection algorithm (PCA-based) for finding dominant strategic dimensions.

7.3 Real-World Example: Housing Entanglement Scenario

A worked example mapping public services failures into DGT observables and ledger evidence. This section shows how to structure FOI/appeal evidence as phenovalue traces that the HDIS network can witness.

7.4 Fault-Tolerant Resolution via Phenomenological Feedback

Active repair flow and decision paths.

7.5 Adaptive Decision Pathways

Algorithmic implementation of adaptive response (Algorithm 2).

Algorithm 2 AdaptiveResponse

```
1: procedure AdaptiveResponse($g, \hat{s}_o, D$)
2:     detect dominant $D_{dom} = \{D \mid E(\hat{s}_o, D) > \theta\}$
3:     for each $D \in D_{dom}$ do
4:         generate counter-strategy $s^{c}_{D}$
5:     end for
6:     combine counters weighted by dominance
7:     return combined strategy
8: end procedure
```

8 Simulation and Verification

8.1 DAG Construction and Traversal Pseudocode

Topological traversal, bounded auxiliary stack and checkpointing notes.

8.2 NFA-Functor Evaluation Loop

Pseudocode for sampling NFA traces, lifting to functors and computing $\mathcal{C}$ estimates.

8.3 Huffman–AVL Compression for State Balancing

Outline of combining Huffman coding heuristics with AVL balancing to reduce state representation entropy while preserving recoverability.

8.4 Complexity Validation Test Cases

Example test harness and expected pass/fail criteria.

8.5 Coherence Score Metrics (Δ-Meaning Scale)

Definition of the witness score used in the registry (scale -12..+12 or 0..1 normalised). Recommended thresholds.

9 Implementation Architecture

9.1 System Layers: IaaS, BaaS, Observer Interfaces

Stack diagram and responsibilities.

9.2 OpenSense Integration and Sensory Phenotyping

Sensor mapping examples (IMU, gaze, skin-response) into phenovalue types.

9.3 Registry and Witness NFT Economy

How coherence receipts can be packaged as signed artefacts for funding applications and audit trails.

9.4 Fault-Tolerant Hot-Swap Engine (SemVerX Resolver)

Resolver pseudocode with Eulerian/Hamiltonian path scoring.

9.5 Prototype: `hdis-d` Daemon and ESP32 Module

Quickstart: clone, build and flash steps; see README in repo.

10 Discussion

10.1 Phenomenological Implications for Human Computation

Ethical stakes, agency and the risk of over-automation.

10.2 Socio-Technical Impact in Public Infrastructure

How witness receipts and coherence scores could change accountability.

10.3 Funding and Accessibility Mitigation via Witness Proofs

Examples of grant/funding documentation derived from ledger evidence.

10.4 Ethical and Legal Dimensions of HDIS

Privacy, consent, and recommended governance model.

11 Conclusion

11.1 Synthesis of DGT–NFA–Functor Model

11.2 Future Research Directions (DGT++ and Quantum Coherence)

11.3 Closing Reflection

References

References

[1] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, 1944.

[2] J. Nash, *Equilibrium points in n-person games*, PNAS, 1950.

[3] N. M. Okpala, *Dimensional Game Theory: DGT (draft)*, OBINexus, 2025.