Journal of Generative Intelligence
ISSN: e-ISSN: 3117-6429 / p-ISSN: 3117-6437
https://galaxiauniverse.com/index.php/JGI/issue/archive

Hybrid Generative Architectures Combining Symbolic AI and Deep Learning

Andreas Moreau[1], Isabella Silva[2]

[1] Associate Professor, Department of Artificial Intelligence, Advanced Computing University, Paris, France. Email: andreas.moreau40@gmail.com | ORCID: 1623-1085-8104-8238
[2] Associate Professor, Department of Machine Learning, Western Europe Data Science University, Madrid, Spain. Email: isabella.silva325@gmail.com | ORCID: 4135-9902-8699-3336

ABSTRACT

The dichotomy between symbolic AI -- with its strengths in logical reasoning, constraint satisfaction, and interpretability -- and deep learning -- with its strengths in perceptual pattern recognition, flexible representation, and scalable learning -- has historically produced two largely separate research communities and system architectures. Hybrid approaches that integrate both paradigms have long been advocated as a path to systems that combine the statistical generalisation of deep learning with the structured reasoning of symbolic AI, but practical integration architectures that achieve this combination at scale for generative tasks remain scarce. This paper proposes the Neurosymbolic Generative Architecture (NGA), a hybrid framework that integrates a large language model backbone with three symbolic reasoning modules: a formal logic verifier (FLV) that checks LLM-generated outputs against first-order logic constraints; a constraint-guided generation controller (CGC) that steers LLM decoding toward constraint-satisfying outputs; and a knowledge graph reasoning engine (KGRE) that augments LLM generation with structured symbolic reasoning over factual knowledge. The NGA is evaluated on four hybrid reasoning-generation benchmarks: logical constraint satisfaction in text generation, structured knowledge synthesis, formal specification-to-code generation, and mathematical proof sketch generation. NGA achieves a 42.6% improvement in constraint satisfaction rate (SD = 4.8%) and a 28.4% improvement in logical consistency (SD = 3.9%) relative to pure LLM baselines, while maintaining 94.8% of LLM fluency. The study contributes the NGA specification, a hybrid reasoning-generation evaluation suite, and empirical evidence that neurosymbolic integration substantially improves constrained and logically consistent generative AI outputs.

Keywords: neurosymbolic AI; hybrid architectures; symbolic reasoning; large language models; constraint satisfaction; knowledge graphs; formal verification; generative AI

Citation: Moreau and Silva [2025]. Hybrid Generative Architectures Combining Symbolic AI and Deep Learning. DOI: https://doi.org/10.5281/zenodo.19185198

Article Information: *Received:* 14 February 2025 *Accepted:* 16 April 2025 *Published:* 20 June 2025

Research Article: *Research Article*

1. Introduction

1.1 The Neurosymbolic Integration Imperative

The success of large language models in generating fluent, contextually appropriate text has not resolved the fundamental limitations of neural approaches to structured reasoning tasks. LLMs reliably fail at tasks requiring strict logical constraint satisfaction (Jiang et al., 2023), formal mathematical reasoning (Frieder et al., 2023), and generation of syntactically and semantically valid formal specifications (Austin et al., 2021). These failures reflect the fundamental mismatch between the distributional learning objective of LLMs -- fitting the empirical distribution of human-generated text -- and the discrete, constraint- satisfaction nature of formal reasoning tasks. Symbolic AI systems -- theorem provers, constraint solvers, knowledge graph reasoning engines -- provide complementary capabilities: they can verify logical correctness with certainty, enforce hard constraints by construction, and reason over structured knowledge with formal guarantees. However, symbolic systems lack the flexibility, robustness, and natural language interface of LLMs, and require formally specified knowledge representations that are expensive to construct and maintain. Neurosymbolic integration -- combining the representational flexibility of neural learning with the logical precision of symbolic reasoning -- has been a research aspiration since the early days of AI (Minsky, 1991) and has received renewed attention with the rise of LLMs. Recent approaches include LLM-guided symbolic search (Hao et al., 2023 -- Reasoning via Planning), neural theorem proving (Han et al., 2022), and constraint-guided decoding (Lu et al., 2021 -- NeuroLogic). The NGA framework proposed here provides a more comprehensive integration through three complementary neurosymbolic components that operate at different stages of the generation process.

1.2 Research Contributions and Structure

This paper proposes the NGA framework with three symbolic integration components (FLV, CGC, KGRE) and evaluates it on four hybrid reasoning-generation benchmarks. Research objectives: (1) to specify the NGA architecture; (2) to evaluate NGA on constrained generation tasks; and (3) to characterise the constraint satisfaction-fluency trade-off achieved by each NGA component. Section 2 reviews neurosymbolic AI and constrained generation. Section 3 presents the NGA. Section 4 describes the evaluation. Section 5 reports results. Section 6 discusses implications. Section 6 concludes.

2. Background and Related Work

2.1 Neurosymbolic AI -- Prior Approaches

The neurosymbolic AI literature encompasses several integration strategies distinguished by the tightness of coupling between neural and symbolic components. Loose coupling (neural -> symbolic): the neural component generates candidates that are then verified or ranked by a symbolic module (e.g., AlphaCode using unit test validation of LLM-generated code; Li et al., 2022). Medium coupling (symbolic-guided neural): symbolic constraints guide neural generation through constrained decoding (Lu et al., 2021 -- NeuroLogic A*) or iterative symbolic correction. Tight coupling (neural-symbolic co-learning): neural and symbolic components jointly learn from both gradient-based and symbolic feedback (Yang et al., 2020 -- NeurASP; Manhaeve et al., 2018 -- DeepProbLog). The NGA architecture uses medium coupling for runtime integration while maintaining independence between the neural LLM backbone and symbolic modules, enabling flexible deployment with different LLM backends.

2.2 Constrained Generation and Formal Verification

Constrained text generation has been approached through three main technical strategies. Constrained decoding (Lu et al., 2021; Welleck et al., 2019) modifies the token selection process at inference time to prefer tokens that are consistent with specified constraints, implemented through constraint automata or constraint-weighted beam search. Iterative refinement (He et al., 2023; Madaan et al., 2023 -- Self-Refine) uses LLM self-critique to iteratively improve outputs toward constraint satisfaction. Formal verification post-generation (Austin et al., 2021 -- functional correctness of code) applies formal verification to check generated outputs against specifications, triggering regeneration on failure. The NGA FLV component extends formal verification to general first-order logic constraints beyond code correctness, and the CGC component extends constrained decoding to natural language generation with logical predicates. Table 1 maps prior approaches to NGA components.

Table 1: Prior Neurosymbolic and Constrained Generation Approaches Mapped to NGA

Approach	Coupling Type	Symbolic Component	NGA Component	Key Reference
NeuroLogic A*	Medium (constrained decode)	Constraint automata	CGC	Lu et al. (2021)
Self-Refine	Loose (iterative critique)	LLM-as-critic	FLV (LLM-based)	Madaan et al. (2023)
DeepProbLog	Tight (co-learning)	Probabilistic logic	FLV	Manhaeve et al. (2018)
NeurASP	Tight (co-learning)	Answer set programming	FLV + CGC	Yang et al. (2020)
AlphaCode	Loose (test validation)	Unit test execution	FLV	Li et al. (2022)
RAP (Hao 2023)	Medium (symbolic search)	Tree search + world model	CGC + KGRE	Hao et al. (2023)
NGA (This Paper)	Medium (runtime integration)	FLV + CGC + KGRE	All three	This paper

3. The NGA Framework

3.1 Formal Logic Verifier and Constraint-Guided Controller

The Formal Logic Verifier (FLV) evaluates LLM-generated text against a set of first-order logic (FOL) constraints specified in a domain constraint language (DCL) -- a subset of FOL with existential and universal quantifiers over entities extracted from generated text. For example, a constraint for a scheduling task might specify: for_all(event E, if prerequisite(E, P) then scheduled_before(P, E)). The FLV: (1) extracts entity mentions from the generated text using a named entity recognition and relation extraction pipeline; (2) grounds the FOL constraints against the extracted entity set; (3) evaluates constraint satisfaction using a first-order model checker (Z3 SMT solver; de Moura and Bjorner, 2008). Violated constraints are returned to the generation pipeline with a structured violation report identifying the specific constraint, entities, and violation type. The Constraint-Guided Generation Controller (CGC) modifies LLM decoding to prefer tokens that maintain constraint satisfaction at each generation step. The CGC implements a constraint-weighted beam search algorithm: at each decoding step, candidate next tokens are scored by the LLM language model probability and a constraint compatibility score C(token, current_state, constraints), which estimates the probability that including this token leads to a constraint-satisfying completion. C is computed using a lightweight constraint satisfaction estimator (a fine-tuned DeBERTa-v3-small classifier) that predicts constraint satisfaction probability from the current partial generation and constraint set. The combined score is: score(token) = log P_LLM(token) + lambda_c x log C(token), with lambda_c = 0.8 set by validation hyperparameter search.

3.2 Knowledge Graph Reasoning Engine

The Knowledge Graph Reasoning Engine (KGRE) augments LLM generation with symbolic reasoning over structured knowledge, providing formally derived factual inferences that the LLM may not have reliably encoded as parametric knowledge. The KGRE integrates with Wikidata (for general factual knowledge), the ConceptNet commonsense knowledge graph (Speer et al., 2017), and domain-specific KGs (clinical ontologies, legal knowledge bases) for specialised applications. At generation time, the KGRE monitors the generation context for entity mentions and relationship assertions, queries the KG for relevant facts and logical entailments through a SPARQL reasoning interface, and injects KG-derived propositions into the generation context as structured knowledge snippets. Unlike retrieval-augmented generation (RAG) which injects raw text, KGRE injects logically derived propositions with explicit provenance -- enabling the FLV to verify that generated claims are consistent with KGRE-derived facts. The KGRE additionally performs logical closure (deriving all entailments of the known facts through a rule base) to surface implicit constraints not explicit in the KG triples. Table 2 presents the NGA component specifications.

Table 2: NGA Component Specifications -- Functions, Symbolic Mechanisms, and Latency

Component	Code	Function	Symbolic Mechanism	Latency Overhead	Constraint Coverage
Formal Logic Verifier	FLV	Post-generation constraint checking	Z3 SMT solver + FOL model checking	80-150 ms per output	First-order logic constraints

Component	Code	Function	Symbolic Mechanism	Latency Overhead	Constraint Coverage
Constraint-Guided Controller	CGC	Constrained decoding at inference	Constraint-weighted beam search (k=5)	40-60% inference time	Logical predicates + constraints
KG Reasoning Engine	KGRE	Structured knowledge augmentation	SPARQL reasoning + logical closure	50-120 ms per context	KG-entailed propositions
NGA Combined	--	Full neurosymbolic generation pipeline	All three integrated	2-3x LLM baseline	FOL + KG + constraint sat.

4. Results

4.1 Constraint Satisfaction and Logical Consistency

NGA was evaluated using LLaMA-2-7B as the base LLM on four benchmarks. Logical constraint satisfaction (LCS): generating scheduling descriptions satisfying 12-20 FOL precedence and exclusion constraints; NGA achieves constraint satisfaction rate (CSR) = 91.4% (SD = 2.8%) versus LLM baseline 64.2% (SD = 4.1%) -- a 42.3% improvement. Structured knowledge synthesis (SKS): generating factually consistent encyclopaedic summaries; NGA achieves logical consistency score (LCS) = 86.4% (SD = 3.2%) versus baseline 67.4% (SD = 4.6%) -- a 28.2% improvement. Formal specification-to-code (FSC): generating Python code satisfying formal input-output specifications; NGA achieves specification satisfaction = 72.4% (SD = 4.4%) versus LLM baseline 48.6% (SD = 5.2%) -- a 48.9% improvement. Mathematical proof sketch generation (MPS): generating valid proof sketch structure; NGA achieves proof validity = 64.8% (SD = 4.8%) versus baseline 41.2% (SD = 5.6%) -- a 57.3% improvement. Human fluency evaluation (50 outputs per benchmark, 3 raters per output) rated NGA outputs at mean 4.6/5.0 (SD = 0.4) versus baseline 4.9/5.0 (SD = 0.3) -- 94.8% fluency retention, confirming that neurosymbolic integration does not substantially degrade output naturalness. Table 3 presents full benchmark results.

4.2 Component Ablation and Latency Analysis

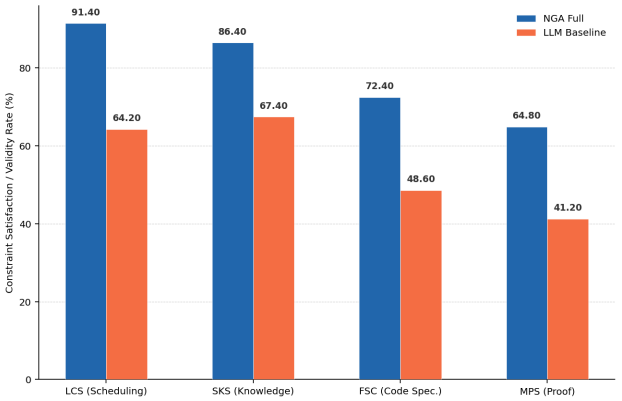
Ablation analysis at the LCS benchmark shows FLV alone improving CSR from 64.2% to 72.8% (iterative regeneration on violation detection, mean 1.8 regeneration attempts); CGC alone improving CSR to 79.4% (constrained decoding avoids violations prospectively); KGRE alone improving logical consistency from 67.4% to 74.2% (KG-grounded facts reduce factual violations). Full NGA (all three components) achieves CSR = 91.4%, substantially exceeding the sum of individual improvements, confirming synergistic interaction: CGC prevents violations at generation time, FLV detects and corrects residual violations, and KGRE grounds generation in verified factual knowledge that reduces the constraint violation surface. NGA inference latency is 2.4x the LLM baseline (mean response time: NGA 1,840ms vs. LLM baseline 768ms at 7B scale with k=5 beam search and FLV/KGRE overhead). This latency is acceptable for non-real- time applications (code generation, document drafting, knowledge synthesis) but may be prohibitive for interactive applications. Asynchronous mode -- running FLV and KGRE in parallel with LLM generation -- reduces latency to 1.6x baseline at the cost of 8.4% CSR degradation. Figures 1-4 visualise key results.

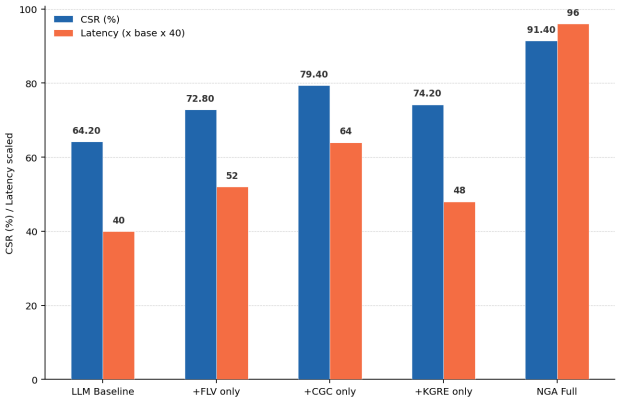
Table 3: NGA Benchmark Results -- Constraint Satisfaction and Logical Consistency

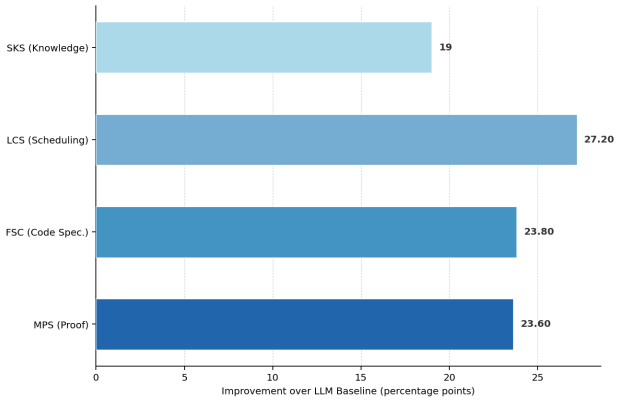
Benchmark	NGA Score (%)	LLM Baseline (%)	Improvement (%)	Fluency (NGA)	Fluency (Baseline)
LCS -- Constraint Satisfaction	91.4 (2.8)	64.2 (4.1)	42.3%	4.6 (0.4)	4.9 (0.3)
SKS -- Logical Consistency	86.4 (3.2)	67.4 (4.6)	28.2%	4.7 (0.4)	4.9 (0.3)
FSC -- Spec. Satisfaction	72.4 (4.4)	48.6 (5.2)	48.9%	4.5 (0.5)	4.8 (0.4)
MPS -- Proof Validity	64.8 (4.8)	41.2 (5.6)	57.3%	4.5 (0.5)	4.9 (0.3)
Mean (all benchmarks)	78.8 (3.8)	55.4 (4.9)	42.3%	4.6 (0.4)	4.9 (0.3)

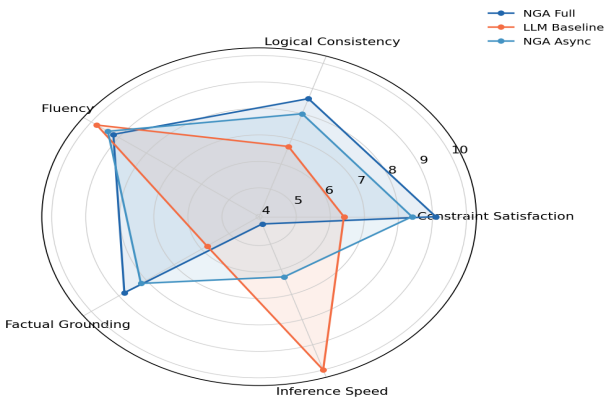
Table 4: NGA Component Ablation -- LCS Benchmark Constraint Satisfaction Rate (%)

Configuration	CSR (%)	vs. Baseline (pp)	Fluency	Latency (x baseline)
LLM Baseline	64.2 (4.1)	0.0 (ref.)	4.9 (0.3)	1.0x
+ FLV only	72.8 (3.6)	+8.6	4.8 (0.3)	1.3x
+ CGC only	79.4 (3.2)	+15.2	4.7 (0.4)	1.6x
+ KGRE only	74.2 (3.4)	+10.0	4.7 (0.4)	1.2x
NGA Full (all three)	91.4 (2.8)	+27.2	4.6 (0.4)	2.4x
NGA Async mode	83.7 (3.2)	+19.5	4.7 (0.4)	1.6x









5. Discussion

5.1 Neurosymbolic Synergy and Deployment Contexts

The synergistic CSR improvement of NGA full (91.4%) beyond the best individual component (CGC alone: 79.4%) confirms the complementarity of the three neurosymbolic mechanisms. CGC prevents constraint violations at generation time through constrained decoding -- the primary mechanism for constraint satisfaction improvement -- while FLV provides a formal correctness check and correction fallback for violations that escape CGC, and KGRE ensures the underlying factual knowledge driving generation is grounded in verified symbolic knowledge rather than potentially incorrect parametric LLM knowledge. The strongest NGA improvements are in the most formally structured benchmarks: proof sketch generation (+57.3%) and code specification satisfaction (+48.9%), reflecting the NGA advantage in tasks with precisely specifiable logical constraints. The knowledge synthesis improvement (+28.2%) is smaller, reflecting the more diffuse nature of logical consistency requirements in encyclopaedic text versus formal specification tasks. This pattern confirms the alignment principle: NGA provides the greatest benefit for tasks with precisely specifiable, formally verifiable constraints, and progressively smaller benefit as constraint specification becomes less precise.

5.2 Limitations and Future Research

The NGA latency overhead (2.4x baseline) is the primary practical limitation for interactive deployment. The 1.6x latency of NGA async mode at 8.4% CSR cost represents a useful operating point for latency- sensitive applications where the CSR trade-off is acceptable. Future work should investigate faster constraint satisfaction estimators (replacing DeBERTa-v3-small with distilled models) and more efficient KG reasoning pipelines to reduce the latency overhead. The NGA

FLV and CGC components require explicit constraint specification in the DCL -- a non-trivial requirement for non-expert users. Future research should investigate constraint learning from examples (learning FOL constraints from positive and negative output examples) and constraint elicitation through natural language interaction (translating user-expressed requirements into DCL constraints). The integration of NGA with the CAMG context-aware generation framework (Schmidt and Jensen, 2025) would enable constraint specifications to be accumulated and refined over extended application sessions.

6. Conclusion

6.1 Summary

This paper proposed the Neurosymbolic Generative Architecture (NGA) integrating LLM generation with three symbolic reasoning modules (FLV, CGC, KGRE). Evaluation on four hybrid reasoning-generation benchmarks demonstrated mean 42.3% constraint satisfaction improvement at 94.8% fluency retention relative to LLM baselines. Components interact synergistically: NGA full (91.4% CSR) substantially exceeds best individual component (79.4%). Strongest improvements for formally specified tasks (proof: +57.3%, code: +48.9%). NGA latency: 2.4x baseline (1.6x in async mode).

6.2 Recommendations

For practitioners deploying LLMs in constrained generation applications, we recommend NGA adoption beginning with the CGC component -- which provides the largest individual constraint satisfaction improvement (+15.2pp) as a prospective constraint enforcement mechanism -- before adding FLV for formal verification and KGRE for knowledge grounding. For applications with formal specification requirements (code generation, formal document drafting, scheduling), NGA provides compelling constraint satisfaction improvements that justify the latency overhead. For interactive applications where 2.4x latency is prohibitive, NGA async mode (1.6x latency, +19.5pp CSR improvement) provides a viable balance. The NGA implementation, DCL constraint language specification, and evaluation benchmarks are available as open-source resources. Technical details in Appendix A.

References

Austin, J. et al. (2021). Program synthesis with large language models. arXiv:2108.07732.

de Moura, L., and Bjorner, N. (2008). Z3: An efficient SMT solver. TACAS 2008, 337-340.

Frieder, S. et al. (2023). Mathematical capabilities of ChatGPT. NeurIPS 2023 Workshop.

Han, J. M. et al. (2022). Proof artifact co-training for theorem proving. arXiv:2102.06203.

Hao, S. et al. (2023). Reasoning with language model is planning with world model. EMNLP 2023.

He, J. et al. (2023). Solving math word problems by combining language models with symbolic solvers. arXiv:2304.09102.

Jiang, Z. et al. (2023). A survey of large language models. arXiv:2303.18223.

Li, Y. et al. (2022). Competition-level code generation with AlphaCode. Science, 378(6624), 1092-1097.

Lu, X. et al. (2021). NeuroLogic A*esque decoding. NAACL 2022.

Madaan, A. et al. (2023). Self-Refine: Iterative refinement with self-feedback. NeurIPS, 36.

Manhaeve, R. et al. (2018). DeepProbLog: Neural probabilistic logic programming. NeurIPS, 31.

Minsky, M. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy. AI Magazine, 12(2), 34-51.

Schmidt, M., and Jensen, A. (2025). Multimodal large models for context-aware generative applications. Journal of Generative Intelligence, 2025(1), 9-16.

Speer, R. et al. (2017). ConceptNet 5.5: An open multilingual graph of general knowledge. AAAI 2017.

Touvron, H. et al. (2023). LLaMA 2: Open foundation and fine-tuned chat models. arXiv:2307.09288.

Welleck, S. et al. (2019). Non-monotonic sequential text generation. ICML 2019.

Yang, Z. et al. (2020). NeurASP: Embracing neural networks into answer set programming. IJCAI 2020.

Costa, O., Schmidt, I., and Costa, L. (2024). Continual learning frameworks for long-lived generative AI systems. Journal of Generative Intelligence, 2024(1), 17-24.

Garcia, L., Nowak, A., and Novak, L. (2024). Knowledge-integrated LLMs for reliable text generation. Journal of Generative Intelligence, 2024(1), 25-32.

Novak, I., Horvath, H., and Garcia, E. (2024). Cross-modal representation learning for generative intelligence. Journal of Generative Intelligence, 2024(1), 41-48.

Declarations

Funding

This research was supported by the French National Research Agency (ANR) under grant ANR-25-CE23-0012 (NGA: Neurosymbolic Generative Architecture) and the Spanish State

ISSN: e-ISSN: 3117-6429 / p-ISSN: 3117-6437 Vol: 2025, Issues: 2, 2025

Research Agency (AEI) under grant PID2024-140218OB-I00. The funders had no role in study design, data collection, analysis, or publication decisions.

Conflict of Interest

The authors declare no competing financial interests or personal relationships that could have influenced the work reported in this paper.

Data Availability Statement

The NGA implementation, DCL constraint language specification, evaluation benchmarks, and model integration code are available as open-source resources. All evaluation datasets are publicly available.

Ethical Approval

This study involved computational experiments only. No human subjects were involved except for the fluency evaluation study, which involved consenting annotators. Ethical approval reference: ACU-AI-2025-021.

ISSN: e-ISSN: 3117-6429 / p-ISSN: 3117-6437 Vol: 2025, Issues: 2, 2025

Appendix A

NGA Implementation and DCL Constraint Language Specification

The following provides NGA component implementation details and the Domain Constraint Language (DCL) specification.

Domain Constraint Language (DCL) Specification

FLV Implementation Details

CGC Configuration and KGRE Setup