Core Emotion Framework (CEF): TS 21 Appendix A — Population Pipeline Pseudocode

Canonical Execution Logic for CEF-KG Population & Integration
Version 1.0 — Phase 4

Author: Jamel Bulgaria
ORCID: 0009-0007-5269-5739
Affiliation: OptimizeYourCapabilities.com
Contact: admin@optimizeyourcapabilities.com

Status: Canonical Appendix (TS-21)

---

0. Purpose and Canonical Position

Appendix A defines the **pseudocode implementation** of the Knowledge Graph Population Pipeline described in TS-21.

It operationalizes:

- extraction
- normalization
- instantiation
- inference enrichment
- constraint enforcement

This appendix introduces **no new emotional constructs**.
It defines the **executable logic** that ensures the CEF-KG remains canonical, identity-preserving, and contamination-free.

---

1. High-Level Pipeline Structure

function Populate_CEF_KG(inputSources):

```
  extracted = extractEntities(inputSources)
  normalized = normalizeEntities(extracted)
  instantiated = instantiateGraph(normalized)
  enriched = applyInference(instantiated)
  validated = enforceConstraints(enriched)
  return validated
```

The pipeline always executes in this order:

1. Extraction
2. Normalization
3. Instantiation
4. Inference Enrichment
5. Constraint Enforcement

---

2. Extraction Stage

```
function extractEntities(sources):
  entities = new EntitySet()

  for each source in sources:
    if conformsToTS18(source):
      entities.add(parseOntology(source))

    if conformsToJSONLD(source):
      entities.add(parseJSONLD(source))

    if conformsToRDF(source):
      entities.add(parseRDF(source))

    if conformsToELSeries(source):
```

```
        entities.add(parseLexicon(source))

    return entities
```

Extraction Rules

- Only canonical sources allowed
- No external emotional constructs
- No non-CEF categories

---

3. Normalization Stage

```
function normalizeEntities(entities):
    state = new NormalizedState()

    state.operatorVector = normalizeOperators(entities)
    state.facetVector = normalizeFacets(entities)
    state.centerVector = normalizeCenters(entities)
    state.modulationMatrix = normalizeModulation(entities)
    state.transitionGraph = normalizeTransitions(entities)
    state.coherenceScalar = computeCoherence(entities)

    return state
```

Normalization Rules

- Identity must be preserved
- Facet ordering must follow TS-11
- No contamination

---

4. Instantiation Stage

```
function instantiateGraph(state):
    graph = new Graph()
```

```
  for each operator in state.operatorVector:
    graph.addNode(operator)

  for each facet in state.facetVector:
    graph.addNode(facet)
    graph.addEdge(facet.belongsToOperator, facet, "hasFacet")

  for each center in state.centerVector:
    graph.addNode(center)

  instantiateTransitions(graph, state.transitionGraph)
  instantiateModulation(graph, state.modulationMatrix)
  instantiateParameters(graph, state)

  return graph
```

Instantiation Rules

- Node types must match TS-18
- Edge types must match TS-20
- No new operators, facets, or centers

---

5. Inference Enrichment Stage

```
function applyInference(graph):
  inferred = copy(graph)

  inferred = inferIdentity(inferred)
  inferred = inferDirectionality(inferred)
  inferred = inferModulation(inferred)
```

```
    inferred = inferStability(inferred)
    inferred = inferPrediction(inferred)
    inferred = inferPlasticity(inferred)
    inferred = inferGovernance(inferred)

    return inferred
```

Inference Rules

- Must follow TS-19 Appendix A
- No inference drift
- No new emotional constructs

---

6. Constraint Enforcement Stage

```
function enforceConstraints(graph):
    if violatesIdentity(graph):
        rollback(graph)

    if violatesDirectionality(graph):
        rollback(graph)

    if violatesModulation(graph):
        rollback(graph)

    if violatesFacetBoundaries(graph):
        rollback(graph)

    if violatesPredictiveRules(graph):
        rollback(graph)
```

```
    if violatesPlasticityRules(graph):
        rollback(graph)

    if violatesGovernanceRules(graph):
        rollback(graph)

    return graph
```

Constraint Rules

- No facet migration
- No operator merging
- No center blending
- No illegal transitions
- No illegal modulation
- No predictive contradictions
- No facet inversion
- No coherence violations

---

7. Update Pipeline (Incremental Updates)

```
function Update_CEF_KG(graph, update):
    if update.type == "parameter":
        applyParameterUpdate(graph, update)

    if update.type == "inference":
        graph = applyInference(graph)

    enforceConstraints(graph)
    return graph
```

Update Rules

- Only parameter updates allowed
- No structural updates
- All updates must pass constraint validation

---

8. Canonical Status

Appendix A is the authoritative pseudocode specification for TS-21.
It defines the executable logic for population, integration, and maintenance of the CEF-KG.

It is subordinate only to:

- Core Essence Document
- TS-1 → TS-21
