Models

Reference Model (Aggregation / Grammar)

ID

- primeID : long

- urn : string

- occurrences : IDOccurrence[]

- embedding : double[]

IDOccurrence : ID

- id : ID

- context : IDOccurrence

- embedding : double[]

Statements:

Data: (IDOccurrence(ID), IDOccurrence(ID), IDOccurrence(ID))

Schema: (ID(IDOccurrence), ID(IDOccurrence), ID(IDOccurrence)

FCA Prime IDs (Embeddings):

Each ID assigned a unique prime number ID. FCA Context / Lattices built upon, for example for a given Data / Schema predicate / arc occurrence role, having the context objects being the statement occurrence subjects and the context attributes the statement occurrence objects, Predicate Context: (Subjects x Objects). For a subject statement occurrence the context is: Subject Context: (Predicates x Objects and for an object statement occurrence role the context is:

Object Context (Subjectx x Predicates).

Graph Model (Alignment, Semantics, Sets / Kinds)

Context : IDOccurrence

Subject : IDOccurrence (Set)

Predicate : IDOccurrence (Set)

Object : IDOccurrence (Set)

Kind : ID

- superKind : Kind

- attributesValues : Tuple<Attribute, Value>[]

SubjectKind : Kind (Predicates intersection Objects)

- occurrences : Subject[]

PredicateKind : Kind (Subjects intersection Predicates)

- occurrences : Predicate[]

ObjectKind : Kind (Predicates intersection Subjects)

- occurrences : Object[]

Statements:

Data: Context(Subject, Predicate, Object)

Schema: Context(SubjectKind, PredicateKind, ObjectKind)

[Sets Diagram]

Activation Model (Activation, DOM / DCI / Actor, Role. Pragmatics)

Instance : IDOccurrence

- id : ID

- label : string

- class : Class

- attributes : Map<string, Instance>

Class : Instance

- id : ID

- label : string

- fields : Map<string, Class>

Context

- roles : Role[]

Role : Class

- dataflow : Dataflow

Dataflow : Context

- role : Role

Interaction

- actor : Actor

Actor : Instance

- interaction : Interaction

Transform

- actor : Actor

Statements:

Data: (Interaction, Actor, Transform)

Schema: (Context, Role, Dataflow)

Helper Services:

Index Service

- FCA / DIDs (ID Creation Time)

- Embeddings Index

- Store / index / query / update index and model’’s embeddings

Naming Service

- Resolve MCP interactions. Content type dispatch.

- Registry IO schema matching MCP resolution.

- Session interaction handler (MCP Resolves client resources / methods to invoke).

Registry Service:

- MCP Resources / contexts.

- Handle sessions / RAG (embeddings)

- IO Representations schema registry (MCPs Structured IO)

Storage Service:

- Internal Graph store (Neo4j / RDF4J)

- Message publish / dispatch. Model Layers events (schemas, content types, topics / queues routing).

- Consumer / Producer sync / bindings. Saga Pattern.

Domain Services:

Augmentation

Consumer

Aggregation

- FCA / Primes Contexts / Lattices.

- DIDs.

- Basic schema inference (IDOccurrences, IDs Statements input).

Alignment

- Type / State inference (Kinds)

- Align / Match Ontologies.

- Order / Dimensional features (Sets).

Activation

- DDD Populated Activation Model.

- Dynamic API Runtime (Conversational State Transfer: COST)

Producer

MCP (LLMs):

Define Reactive Streams Functional Processing APIs: MCP’s tools, prompts templates and resources). Content type dispatch. MCP Calling client methods / resources. Provider (Helper) Services: define APIs. Embeddings.

Tool (Reactive Microservice) interfaces (Functional APIs):

- inputs: consume streams messages matching pattern / content type.

- core tool logic: consume / produce from server. Callbacks

- produces: publish (registry) inputs / produces signatures. return / publish type structured outputs.