MetaModel:

Arrangement of layered typed CSPO quads statements in which each CSPO role / type plays the role of "aggregating" previous layer abstract knowledge into more concrete aggregated contexts (statements) instances until a "reference model" CSPO type arrangement is achieved, which is the root of all other layers hierarchy.

The purpose of this is to achieve some ontology matching capabilities over an upper abstraction set of layers and to enable a Functional Knowledge Base Interaction APIs integration / virtualization overlay for matching and consumption of distributed datasets via endpoints dataflows.

**Reference Model:**

ID : URL;

Occurrence: Context;

Context : ID (Context / ID : intension, Object / Occurrence, Sign / Kind / Metaclass / Attribute, Value / Role / Class : extension);

Root of MetaModel hierarchy.

**Notation:**

[LayerType] : [LayerSuperType] ([ContextType], [SubjectType], [PredicateType], [ObjectType]);

**Layers:**

Resource : Statement (Resource, Resource, Resource, Resource);

Statement : Role (Statement: c, Resource, Resource, Resource);

Role / Class : Kind (Role: b, Statement, Resource / Attribute, Resource / Value);

Role / Class aggregating CSPO Resource (IDs) sharing Attributes for their Objects / Values.

Kind / Metaclass : Relation (Kind: a, Role, Statement, Resource);

Kind: Aggregated similar Roles occurring as Resources (Object) in Statements (Predicate).

Relation / Entity : Mapping (Relation, Kind, Role, Statement: c);

Rel type (Relationship) instance / bindings. An Entity (Relation: intension) and their Statements for its Kind / Role occurrences (occurrences: kinds / roles Relation plays in statements. Matching. Object: extension). Data (DCI)[1].

Mapping : Relationship (Mapping, Relation, Kind, Role: b);

Rel players types / bindings scenarios. Information. Interaction (DCI)[1]. Mapping Role and Relation Kind: dataflow promoted types / order: relationships players domain / range. Entity alignment.

Relationship : Value (Relationship, Mapping, Relation, Kind: a);

Rel type declaration, player types. Knowledge. Context (DCI)[1].

Value : Sign (Value, Relationship, Mapping, Relation);

Sign : Object (Sign, Value, Relationship, Mapping);

Object : Context (Object, Sign, Value, Relationship);

Context : ID (Context, Object, Sign, Value);

Reference Model:

Root of MetaModel hierarchy.

**Matching / Relations / Attributes:**

One of the intentions of having all this layered infraestructure is to be able to inspect "relations", being them "reified" into a Relationship construct, or being them single attributes and values for a subject enabling the possibility of "align" one into another for ontology matching purposes.

(a, b, c: Kind, Role, Statement): Reified Rel. to / from expanded Attributes / Values. Matching / roles (intension / extension).

Context DOM: parent / child; previous / next siblings; attribute / value (determined by CSPO roles). Class / instance DOM relation for parent / children layers instances.

**Ontologies:**

Context layers instances. Levels. Example: Dimensional ontology. Ontologies should be able to be built upon Reference Model layer CSPO types arrangements.

Dimension, Unit, Measure, Value.

Axis, Behavior, Flow (state change), etc.

Primitives: dimensional upper ontology. In / Out, Prev / Next, Pick / Drop, etc. Opposites. State change (current). Events, state flows. Marriage example.

**Message Dataflow:**

Relationship, Mapping, Relation streams / signatures. Messages: Context instances. Functional Knowledge Base Interaction APIs.

Aggregation: Browse / Transform.

Alignment: Inference.

Activation: Dataflow type (signatures). Message dispatch (domain / range ordered). Aggregation.

**Relations:**

Inference. Relation types: transitive, reflexive, simetric. Campo, alcance, dominio, rango, transform / function: infer / aggregate. Context functor / monad.

Inputs: (Context / Relation, PK, column, value);

Inputs: (Infer S Kind / Role, S, P, O);

Inputs: aggregate occurrences. Statement Context for each SPO as Occurrence with corresponding Attribute / Value (S: PO, O: SP, P: SO, etc.).

Inputs (infer rels): Part / Whole. SPO / OPS. Attribute / Value.

Inputs (infer rels): Containment. SPO / SPO. Parent / Children. Occurrences of Contexts of same Context layers (inherited Contexts). Example: (Mapping, Mapping) for (Relationship, Mapping). Super / Sub type Contexts instances relationships.

Inputs (infer rels): Order. SO Ps Domain / Range.

Input (infer rels): Event. Prev / Next state change. Type promotion.

**Reactive Functional Reified Metamodel:**

Transforms: Match Selectors. Hierarchy polymorphism. Contexts streams. Browse Metamodel. Context, Subject Selectors.

Transforms: Templates. Context instances (CSs) declaration / augmentation (POs) Selectors. Metamodel activation. Predicate / Object Selectors.

Encode Match / Template as Context. CS: Match, PO: Template Selectors. Apply Templates (role bindings / prompts) declaration / augmentation activation to matching selected CS streams.

Augmentation: Aggregation, Alignment, Activation Reified Match / Template dataflows. Reactive Model instances Match / Template dataflows.

Selectors:

Apply Role to Statement : Statement / Statement to Role : Role.

Apply Kind to Role : Role / Role to Kind : Kind.

Apply Relation to Kind : Kind / Kind to Relation : Relation.

Apply Mapping to Relation : Relation / Relation to Mapping : Mapping.

Apply Relationship to Mapping : Mapping / Mapping to Relationship : Relationship.

Apply Context to Relationship : Relationship / Relationship to Context : Context.

**MetaModel / Backend:**

Reference Model / Occurrences annotations matrix:

Attachment: show an example of a fully expanded set of last model layer (Relation) and its corresponding occurrences property graph annotations. Previous model layers may be annotated accordingly in respect to their ability to aggregate more abstract contexts (Resource, Context) properties.

Reference Model / Occurrences annotations matrix:

Lattice / FCA: Contexts / Resources. Objects / Attributes (Contexts instances axes). (X, Y): Z (for corresponding pair types / functional transforms).

Statements and annotations: FCA Lattice / FCA Contexts / Attributes. Objects / Attributes (Contexts instances axes). (X, Y): Z (for corresponding pair types / functional transforms). Typed calculus in FCA context development. Sets.

IDs / bitstring encodings. Algebraic / arithmetic activation flows / templates / transforms metadata encoded selectors. Vector Space Model quads polygon embeddings.

Interfaces:

ID : URL;

Context: ID;

Object : Context;

Sign : Object;

Value : Sign;

Labeled Property Graph annotations example. Augments Reference Model. Statement example, Statement context aggregates SPO annotations (Statement occurrences data in other Reference Model layers contexts):

Context (Context, Object, Sign, Value);

Resource : Value (Resource, Resource, Resource, Resource);

Statement : Resource (Statement, Resource, Resource, Resource);

Role : Statement (Statement, Role, Resource, Resource);

Kind : Role (Statement, Role, Kind, Resource);

Relation : Kind (Statement, Role, Kind, Relation);

Relationship : Relation (Relationship, Role, Kind, Relation); \*

\*: Relationship: Aggregated Relation Statement Relation (Object) Roles / Kinds.

Intension / Extension: S / O.

sub / super hiers, containment: P / O.

Relation reification: Relation statement object: relation instance. a: Role / b: Kind: relation ends. (a): Role reifying rel attrs / values. (b): Kind Resource reifying rel subject (rel players).

**Model:**

Message Events Bus.

Context Monad & type hierarchy (AST). CSPO parameterized types & aggregation (layers hierarchies specializations).

DOM:

Layer (Contexts): events producer / consumer (streams observer / observable).

DOM Parsing MetaModel: aggregate occurrences containment / hierarchies. Layers: subtype / supertype browse parent / children / siblings (order) and Attributes / Values relations.

Context::matchFilter(arg : Context): signatures / kinds stream predicate.

Context::applyMap(arg : Context): apply updates (CS Contexts / PO CRUD) matching filter predicate Context. Fires event bus messages.

Matching applies to meta-model signatures (internal aggregation, alignment and activation augmentations) and to domain / actual models signatures. Render new Attributes / Values and CSPO statements.

DDD: Declarative AST / Dataflow VM. Runtime. Encodings (layers / messages / activations). Event sourcing backends (Blockchain). Patterns (DCI).

RDFS / OWL / Graph Backend: Reified Metamodel. Labeled Property Graph. APIs.

Functional Meta Model Context Layers. Reactive streams. Match / Template Selectors. API.

Stream Resources Connectors (P2P Connector Bundles Context I/O). Dataflow. Backends / Augmentations / Endpoints model layers. APIs (reactive / events).

Layered abstraction levels streams options / menu semantics (REST). Forms / Flows. Browse, match / transforms high level APIs. HATEOAS CRUD / Flows.

Patterns. Input formats. Sample data.

**FCA / Concept Lattices**

Model context statements of an upper ontology and occurrence statements for each ontology concept occurrences using FCA (Formal Concept Analysis) and rules from a pseudo-grammar.

Context statements / Occurrence statements: Grammars. Concepts / objects hierarchies: CSPO statements concept types / kind rules / terminal instances. Productions: concept types / kind rules / terminal instances mappings / flows.

Aggregate kind rules (grammar) into context statements / Occurrence statements: polymorphically, Kind context applies to all Resource hierarchy (all lattices).

Contexts parsing: monadic parser combinators / monadic AST. Recognize context types from (surrounding) reified kind types / rules (link grammar).

Rules are of the form:

(TypeA, TypeB) > AggregatedKindResources;

For example, in Relation lattice:

(someRoleA, someKindB) > AggregatedRelationResources;

Dispatch: Model (lattices) observer / observable (streams / functional) of grammar (contexts case match) events.

Lattice (FCA Contexts) population and Augmentation:

Base layer: Context. Resource, Statement, Role, Kind, Relation layers.

Input layer: case match layer grammar type signature. Reactive dataflow dispatch inputs top-down / bottom-up contexts layers hierarchy (lower hierarchy layers polymorphically materializes upper hierarchy layers) for Augmentation.

Lower hierarchy layers contexts mapping / function transforms into next upper layer context (example: Roles to Kinds). TBD.

Augmentation:

Activation: Layer receives matching context message.

Aggregation: Layer resolves productions of grammar rules for context message.

Alignment: Resolution of relevant knowledge and input message. Emits output message.

Notes:

Context occurrences statements (i.e.: Statement in Kind context). Occurrence contexts: (S, S); (P, P); (O, O);

FCA: Lattice. Ordered Sets. Intension / Extension.

To Do:

Model Order. Axes: dimensions, units, measures. Events (measures / relations / state boundaries in data / information / knowledge levels: price, price at moment in time, variation, tendence predictions, idem for distances, etc.).

Hierarchies: metaclass / role, class, instance, occurrence (parent, children, previous, next, attribute, value). Encoding. Comparisons. Functional traversal (streams).

Dimensional contexts: Contexts from Occurrences contexts statements. Dimensional contexts: Events (attributes). Order relations assertions by context occurrences hierarchy domain / range, set / superset attributes relations.

(Mapping, Kind, Role, Statement);

Event (Dimensional context attributes): (Mapping / unit / class, Mapping super / parent / dimension / metaclass, Kind unit / measure / occurrence, Role measure / value instance);

Model Application domains: upper ontology (Behavior, Flow, etc.) encoded in meta model and specialization levels for domains contexts. Declarative abstractions ontology for application design: discovery, alignment and matching for services renderings and integrations.