Sets (Quads):

Sets (Resources, Subjects, Predicates, Objects, SubjectKinds, PredicateKinds, ObjectKinds, Statements: Mappings / Transforms) abstraction for representing Augmented RDF Graphs.

Augmentations:

Inference mechanisms for obtaining metadata from input statements and augmenting, for example, type information and models schema and transforms.

Data (Activation), Schema (Aggregation), Behavior (Alignment):

Data Activation

Schema Aggregation

Behavior Alignment

Functional Sets Relations:

Classes (Sets) domain hierarchy:

OntResource
Subject : OntResource
Predicate : OntResource
Object : OntResource
Kind : OntResource
SubjectKind : Kind
PredicateKind : Kind
ObjectKind : Kind
Statement : Kinds, SPOs
Context : Kinds, SPO
Mappings : Kinds, SPO

Functor / Category: Resource Monad (of OntResource hierarchy). Dynamic typing DOM / DTOs Kinds members.

OntResource: Uniform Resource domain category interface:

getSet
getKind
getOntResource
getResource (this)
getContext
getOccurrence
getAttribute
getValue
getQuadContext
getQuadSubject
getQuadPredicate
getQuadObject

Domain Object Members, i.e.: getSubjectKind.

Class, Metaclass, Instance, Context, Occurrence, Role. Encoding: Functional Mappings / Transforms. Order. Relations, Data Flow Roles.

Domain Model Object Hierarchy:
ClassName :: (aggregatingClass, subject / instance, attribute / predicate, value / object);

SPO/Kinds Set: Contexts (metaclass, class, instance, context, occurrence, role, etc.)

OntResource model Quads hierarchy:

OntResource: Universe Set.
(OntResource, OntResource, OntResource, OntResource);

Subjects : OntResource
(SubjectKind, Subject, Predicate, Object);

Predicates : OntResource
(PredicateKind, Subject, Predicate, Object);

Objects : OntResource
(ObjectKind, Subject, Predicate, Object);

SubjectKind (SK) : Subject. Predicate / Object Intersection.
(Statement, SubjectKind, Predicate, Object);

PredicateKind (PK) : Predicate. Subject / Object intersection:
(Statement, Subject, PredicateKind, Object);

ObjectKind (OK) : Object. Predicate / Subject intersection. Occurring.
(Statement, Predicate, Subject, ObjectKind);

Statements : Kinds / SPOs
(Kinds, Resource, Resource, Resource);

Contexts : Kinds / SPOs
(Statement, Kind, Kind, Kind);

Mappings : Kinds / SPOs.
(Context, Transform, Transform, Transform);

Transform : Kinds / SPOs
(Mapping, Resource, ResourceMember / Op, Value);

Augmentations:
Contexts matching Statements applied to aggregated Mapping Context Transforms.
Apply Mappings Transforms. Transform Values Statement (Transform interface reifies Value as Statement Resource)

Implement Functional APIs:

Activation (Data)

Aggregation (Schema)

Alignment (Behavior)

Implement recursion, aggregation, order, data flow, activation, alignment.

Domain Type Hierarchy: Reification, Resource Functor Transforms Domains: subtypes transforms wrapped compatible with results wrapped types by inheritance.

Type Inference: Kinds (Classes):

Aggregate same Attributes occurrences for sets of Resources sharing same Attributes. Activate Context Transforms Kinds. Activate Kinds Resources Statements.

Wrapped Types (Kinds) Inputs Inference / Matching. Wrappers contains Wrapped CSPO Role Resources. Functional Flow into Occurrences, Attributes, Values.

Encodings. Representations: Instances / Literals Encoding. URNs. Resolution: sameAs Mappings / Parsing. Occurrence / Occurring domainOf / rangeOf Type Inference.

Model Kinds: Model Reified.
Domains Kinds: From inputs.
Reified Model Resource Kinds.
Functional: Monads (wrappers types / wrapped types inference). Kinds Domain Flow (Mappings):

DOM Resources: dynamic object model / kinds.

Sample Statements:
(Dimension, Measure, Unit, Value);
(Relationship, Relation, Role, Player);
(Time, 1h, mins, 60m);
(Working, 1h, USD, 40);
(Working, 160h, USD, ?);

Model API:

Inputs / API:

I/O Normal Form: Statement
Service Facade. Functional Data Flow: Matching Mapping Transform: Statements. REST HATEOAS URNs:

I/O Statement:
(Context / Class, Instance, Attribute, Value);

Data Flow: Service Facade API:

REST Data Flow: Services Facade URN request / response HATEOAS flow.
Transform::Mapping::Statement::Kind::Resource;
Resource::Kind::Statement::Mapping::Transform;

Sets Resources REST HATEOAS / Data Flow IO Model Statements:

(Transform, Mapping, Statement, Kind);

Functional Data Flow:

Transform::Mapping::Statement::Kind::Resource;
Resource::Kind::Statement::Mapping::Transform;

Encoding.

Augmentations:

RDF Backend. Event sourcing (bus) saga pattern. Publish / Subscribe. Connectors.

Data Matching: Activation.

Schema Matching: Aggregation.

Behavior Matching: Alignment.

Activation (Data Matching):

RDF Quads Parsing from events sourcing events bus:

(Class, Instance, Attribute, Value);

Populate SPOs / Statements / Kinds / Mappings / Transforms Quads Wrappers Sets Objects for Aggregation.

Ontology Matching: Resources Kinds Matching. Merge same URNs.

Aggregation (Schema Matching):

Aggregation. Quads CSPOs / Attributes / Values. Handle recursion. Functional Transforms Context: subjectKind::subject::subjectKind (same subjectKind).

Schema Matching: Aggregation Kinds Matching.

Resources aggregate into Kinds. Kinds aggregate into Statements, Statements aggregate into Mappings. Mappings aggregate into Transforms. Hierarchy aligns Wrapper types reification.

Quad Wrappers (Resource hierarchy) wraps aggregated occurrence of wrapped Quad Type. Wrapped Quad Type: Kind. Wrapped: DOM / DTO of Kind members.

Alignment (Behavior Matching):

Resources Reification: Kinds, Statements, Mappings, Transforms reified. Reified Resources aggregates aligned into Transform Wrapped Quads:

(Kind, Statement, Mappings, Transform);

HATEOAS Functional Browsing. RDF Model Serialization.

Behavior Matching: Transform Quad Kinds Matching.

- Notes:
- URN : Resource (alignments). Primitives.
- Resource : Root Category. URN : Source / Surrogate Key / Crafted. Naming / Encodings (below).
- Ontology alignments: Data / Schema / Behavior Augmentations. Model / Schema / Upper / Domains: purposes / gestures (MVC / DCI Mappings / Transforms) layers. Example:
- Occurring / Context (Statements / Kinds)
- Roles (Metaclass, Class, Occurrence, Context, Role)
- MVC / DCI Mappings / Transforms. Example: Forms, Purpose, Gestures, Actors, Roles. Data / Schema / Behavior alignment.
- ESB: Endpoints, Features, Interfaces, Service Process Description / Discovery. Reactive Events Subscriptions. HATEOAS Endpoints "autowiring".
- BPM: Process, Steps, Flows, etc.
- Augmented Actionable (Process Flows, Items Activation) CMS. Browser: HATEOAS Protocol / APIs / Augmentations. Inferred / Reified / Resolvable Data Flows. Designer: Model Pallete. Declarative core / domains types / instances browsing / discovery "wiring".
- Graph Reified Grammars (upper). Contexts / Mappings. Terminal / Non Terminal. Rules / Productions. Mappings / Transform: browse grammar, rules, productions:
- (Rule, Context, lhs, rhs)
- Naming: Kinds / URNs Addressable Encodings. Parsing: URNs Encoded Functional Distributed Resource Resolution. Data Flow Transform / Mappings: Embedded Productions: Augmentations. NLP / NER. Ontology Matching: URN Class Transforms.
- Graph Embeddings: ML Backend Services (ML Predictions Augments Mappings / Transforms). Encodings (Naming).
- Encoding: Deep ML Embeddings. Data: classification, Schema: clustering, Behavior: regression.
- Naming: Auto Encoders. Semantic Hashing. Resources Mappings / Transforms Reified Maps / Tables. Keys / Values Resource Hashing / Resolution Functions: Contextual to Functional Environment State: Mappings Flows / Wrapped State.
- Naming: Augmentations. Contextual Hash Enabled: Functional Mapping Flows Map / Table Encoded / Resolved. Functional Relations: Ontology Matching / Aggregation / Inferences by Hash Encoded Metadata / Transforms Resolutions.
- Clients / Browsers: Peers. Protocol: Reactive Dialogs Prompts. Events. Distributed Data, Schema, Behavior Core Model Statements Encoded I/O: Layers Sync / Augmentation of Knowledge requested from each Peer(s) as Model inputs given resolution of Dialog (Subscriptions) event sourcing state. MVC / DCI Distributed State Transforms / Mappings. Augmented Peer(s) Models: updated View State (flows) / Mappings / Transforms. Rendezvous Peer Role. Local Peer: APIs for local / remote views (MVC / DCI) views (Web, REST) Rendering.
- (...)
-