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 (Statement), Schema (Mapping), Behavior (Transform):

Aggregation

Activation

Alignment

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;

Functional Sets Relations. Mappings / Transforms.

Sets (Wrappers): Parsed input Statements Resources instances are wrapped into Resource Monad Categories Statements:

Resource, CSPO Resources wrappers, Kinds Resources wrappers and Statement, Mapping, Transform wrappers.

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;

Wrappers Sets:

CSPO Statement Form: for Transform Augmentations.

Functional Template Form: for aggregation inference mechanisms.

Wrapper Statements:

CSPO Statements Form:

Resources: Universe Set.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource); Normal Form.

Subjects : Resource

(SubjectKind, Subject, Attribute : Predicate, Value : Object);

Predicates : Resource

(PredicateKind, Attribute : Resource Subject, Predicate, Value : Object);

Objects : Resource

(ObjectKind, Attribute : Resource Predicate, Value : Resource Subject, Object);

SubjectKind (SK) : Subject. Predicate / Object Intersection.

(Statement, SubjectKind, Attribute : Predicate, Value : Object);

PredicateKind (PK) : Predicate. Subject / Object intersection:

(Statement, Attribute : Subject, PredicateKind, Value : Object);

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

(Statement, Attribute : Predicate, Value : Subject, ObjectKind);

Statement / Mapping / Transform: Subject / Predicate / Object intersection:

Dual Occurrence of Transform (Mapping) / Occurring of Mapping (Transform) for Statements. Templates: apply Transform Matching Mapping Statements.

Transforms : Mappings

(Context : Mapping, Occurrence : Resource, Attribute : Resource, Value : Resource);

Mappings : Statements

(Context : Transform, Occurrence : Resource, Attribute : Resource, Value : Resource);

Statements : Kinds

(Context : Mapping, Occurrence : Resource, Attribute : Resource, Value : Resource);

Wrapper Statements:

Functional Template Form:

Resources: Universe Set.

(Context : Resource, Occurrence : Resource, Attribute : Resource, Value : Resource); Normal Form.

Subjects:

(Subject, Occurrence : SubjectKind, Attribute : Resource P, Value : Resource O);

Predicates:

(Predicate, Occurrence: PredicateKind, Attribute : Resource S, Value : Resource O);

Objects: (Object, Occurrence : ObjectKind, Attribute : Resource P,  Value : Resource S);

SubjectKind (SK) : Subject. Predicate / Object Intersection. Occurrence:

(Context : SubjectKind, Occurrence : Statement, Attribute : Predicate, Value : Object);

PredicateKind (PK) : Predicate. Subject / Object intersection:

(Context : PredicateKind, Occurrence : Statement, Attribute : Subject, Value : Object);

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

(Context : ObjectKind, Occurrence : Statement, Attribute : Subject, Value : Predicate);

Statement / Mapping / Transform: Subject / Predicate / Object intersection:

Dual Occurrence of Transform (Mapping) / Occurring of Mapping (Transform) for Statements. Templates: apply Transform Matching Mapping Statements.

Transforms : Mapping.

(Context : Transform, Occurrence : Mapping, Attribute : Resource T, Value : Resource : U);

Mappings : Statement.

(Context : Mapping, Occurrence : Transform, Attribute : Resource T, Value : Resource U);

Statements : Kinds

(Context : Statement, Occurrence : Mapping, Attribute : Resource, Value : Resource);

Wrappers Type Hierarchy: Reification, Functors Transforms Domains (subtypes transforms wrappers compatible with results wrapper 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, ?);

Augmentations:

Encoding: Naming. Model Resource URNs. Registry resolves names to parsed wrapper objects. Restore / parse serialized objects model.

RDF serialized model: URNs with metadata to encode / resolve Objects Resources Model (Registry Service).

Naming: Dictionary. Relations. URN IDs operations. Index: Query occurrences and aggregations.

SAIL: Registry resolves encoded models objects by name. Passivates objects into models graph.

Alignment (Behavior Matching):

SAIL: Alignment. Reified Resources aggregates aligned Wrapped Quads:

(Kind, Statement, Mappings, Transform);

Data: Statement; Matches Kinds / SPO: Mapping : Statements matching context.

Schema: Mapping; Matches Kinds / SPO: Transform : Mappings matching context.

Behavior: Transform; Matches Kinds / SPO: Transforms : Mapping Statements matching context.

Alignment API: Functional Functors / Transforms data flows.

Activation (Schema Matching):

SAIL: Activation. Resources Reification: Kinds, Statements, Mappings, Transforms reified

Aggregation (Data Matching)::

SAIL: Aggregation. Quads CSPOs / Attributes / Values.

Aggregation renders Resources into Functional Template Form (Statements / Mappings / Transforms updated) for Attribute / Value aggregations inference.

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

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

* Aggregate Resources for Attribute / Values. Statement: SPO Resource
* Aggregate SPO Resources:
* Subjects: (Subject, Occurrence : SubjectKind, Attribute : Resource P, Value : Resource O); Kinds
* Aggregate Kinds:
* (Context : SubjectKind, Occurrence : Statement, Attribute : Predicate, Value : Object); Statements
* Aggregate Statements:
* (Context : Statement, Occurrence : Mapping, Attribute : Resource, Value : Resource); Mappings
* Aggregate Mappings.
* (Context : Mapping: Occurrence : Transform, Attribute : Resource T, Value : Resource U); Transform
* Aggregate Transforms:
* (Context : Transform, Occurrence : Mapping, Attribute : Resource T, Value : Resource : U); Mapping
* Template Transforms (Mapping Roles): TO DO
* Aggregation: Statements for each Context Occurrence Attribute / Value. Mapping: for each matching Attribute / Value apply Transform, render Statement apply Attribute / Value Transform. Context: Transform / Class. Occurrence: Subject. Normal Form. Transform wrapped: Context / Class, Occurrence wrapped: Subject.
* Augmentations: Activation (Schema), Alignment (Data), Aggregation (Behavior) Matching (Mapping Function) results: Template Transforms (noop, merge, add); Transforms Flow State: listening for Matching Inputs.
* Aggregation: Statements for each Context Occurrence Attribute / Value. Mapping: for each matching Attribute / Value apply Transform, render Statement apply Attribute / Value Transform. Context: Transform / Class. Occurrence: Subject. Normal Form. Transform wrapped: Context / Class, Occurrence wrapped: Subject.
* Aggregation: Statements for each Context Occurrence Attribute / Value. Mapping: for each matching Attribute / Value apply Transform, render Statement apply Attribute / Value Transform. Context: Transform / Class. Occurrence: Subject. Normal Form. Transform wrapped: Context / Class, Occurrence wrapped: Subject.
* Alignment: Aggregate Resources Context Occurrences, Attributes, Values for Resource, Kinds, Statements, Mapings, Transforms Resources from Statement, Mapping, Transforms occurrences / occurring. Positional Roles: Functional Resources Roles Reification. Wrapper Types. Aggregation: Matching.
* Occurrence Object Member of Subject as Transform / Function Role.
* Occurring Subject Member of Object as Mapping / Function Role.
* Model Aggregation / Expansion (Augmentations match / apply) of Mappings / Transforms Core Statements
* Subscriptions: domain / range.
* Aggregation: Transforms Reified in Layers Contexts. Pattern Matching Template Layeresolved.ed
* (Wrapper, Wrapped, Mapping, Transform);

Functional APIs:

* SPOs, Kinds, Statements Contexts. Functional flow.
* Sets API. Wrapper::getSet
* Quad: Wrapper Wrapped CSPOs / Functional / OntResource. Root interface.
* Quads Wrappers: Resource hierarchy (Resources, Kinds, Statements, Mappings, Transforms). Wrapper Sets: Corresponding Class
* Quad CSPO wrapped Resources (Quads) relative to context: SubjectKind Quad instance into Subject Quad relative to occurrence context (SubjectKind Occurrence for Subject Attributes / Values. Occurrence Resource Role.
* Quad: CSPO Resources Set Wrapped Quad Functional API (Quad interface): getSet (Wrapper) / getKind (Wrapped) / getOntResource / getResource (this) / getContext / getOccurrence / getAttribute / getValue / getQuadContext / getQuadSubject / getQuadPredicate / getQuadObject. Resource hierarchy mapping getters and setters (i.e.: Attribute / Predicate). Aggregation Transforms.
* Resources wrap Quads / OntResources (Source URN / matching: wrapper / wrapped ID encodings).
* IDs: scheme:wrapperType:wrapped Kind:wrappedID:occurrenceID..
* Class, Metaclass, Instance, Context, Occurrence, Role. Encoding: Functional Mappings / Transforms. Order. Relations, Data Flow Roles.
* Inputs: (Class, Instance, Attribute, Value): Aggregate: Functional Form implemented in Quad interface.
* Resources aggregate into Kinds. Kinds aggregate into Statements, Statements aggregate into Mappings. Mappings aggregate into Transforms. Hierarchy aligns Wrapper types.
* Quad Wrappers (Resource hierarchy) wraps aggregated occurrence of wrapped Quad. Wrapped Quad Type: Kind. Wrapper: DOM / DTO of Kind members.
* Resources wrap Quads (and OntResources: Source URN / matching: wrapper / wrapped ID encodings.
* 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). 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.
* (...)