* SAIL Layers (Storage and Inference Backends):
* Ternary Backend SAIL
* Sets Backend SAIL
* Statements / Quads Backend SAIL
* Services / Layers Quads Protocol.
* Layers Backend Components (Quads I/O) Services:
* Quads: Services / Layers Protocol
* CDI: Signatures Functional Dataflow Injection / Bus Resolution.
* Aggregation Statements Types:,
* (Kinds / Types, Object / Resource, Predicate / Arc, Object / Resource);
* (Object / Resource, Kinds / Types, Predicate / Arc, Object / Resource);
* Aggregation Statements Types:
* (Kinds / Types, Object / Resource / Predicate / Arc, Object / Resource);
* (Subject / Context, Kinds / Types, Predicate / Arc, Object / Resource);
* Reification. Infer Kinds / Types vía repeated matching of Statements Attributes occurrences. Quads Encoded. Attribute relation: Mapping from and edge of whim an Arc / Predicate is source.
* Encoding: Reification of Statement types to canonical form:
* (Context, Subject, Attribute, Value);
* Aggregate / de Aggregate mappings expansions.
* Operations: Monads, Transforms, Dataflow. Templates / Mappings for processing representations. TODO.
* Data Matching Services (Nodes / Resources)
* MVC: Model; Index Service
* DCI: Data; Index Service;
* Schema Matching Services (Kinds / Types)
* MVC: View (flows / prompts); Registry Service;
* DCI: Context; Registry Service;
* Behavior Matching Services (Predicates / Arcs)
* MVC: Controller; Naming Service;
* DCI: Interaction; Naming Service;
* Augmentations:
* Alignment (Data Matching)
* Activation (Schema Matching)
* Aggregation (Behavior Matching)
* Index Service
* Naming Service
* Registry Service
* DDD (Meta Circular Interpreter) : Code as Data.
* Prompts / Flows: View Forms.
* Meta Model : Dataflow
* Use Cases
* Kind / Type Sets / Quads: Resource : Statement
* Kind / Type
* Statement:
* (Context : Statement, Subject : Resource, Attribute : Kind, Value : Resource);
* Value Resource according Kind (Roles) Statement Subject occurrence.
* Sets: de aggregated Statements. Reification. Layers (Models) expands Statements for each Kind Attribute.
* Graph:
* (Context : Type, Subject : Node, Attribute : Arc, Value : Node);
* Kind / Type: Aggregated Attribute / Arc occurrences. Type Inference.
* Type Context Arcs Expansion. Augmentations.
* Translation Layers:
* Ternary Translation Layers:
* Input: Parse Quads : Emits Graph Statements Roles.
* Augmentations
* Output: Visit Ternary Mappings. Emit Statements.
* Sets Translation Layers:
* Input: Parse Quads : Emits Graph Statements Roles.
* Augmentations
* Output: Visit Ternary Mappings. Emit Statements.
* Quads Translation Layers:
* Input: Parse Quads : Emits Graph Statements Roles.
* Augmentations
* Output: Visit Ternary Mappings. Emit Statements.
* Ternary Translation Layers:
* Augmentations / Matching.
* Ternary: (Sign, Concept, Value);
* Resource Roles:Ontology Matching: Wrapper / Wrapped Types.
* Data, Schema, Behavior Monads / Functors.
* Object, Concept, Value Roles.
* (object) : (type) : (arc)Arc : Type object occurrences.
* Property Graph Nodes / Arcs Quads Augmentation I/O: Rendering / Parse. SAIL Backend specific encoding / transform.
* Encoding / Matching:
* Functional Primitives: (Matching). Graph Shapes Model. Layers. Example: reify / render / match "uncle" relation / "marriage" situation from graph statements.
* Primitive slots template Quads (Roles) populating "shapes" matching promps / flows / state transitions.
* Metaclass
* Class
* Instance
* Context
* Role
* Occurrence
* Primitive Relations.
* Functional Transforms: Browse State / Traversal. Navigational Context (Primitives State) Builder of Mappings (Templates).
* Augmentations: Matchings.
* Abstract Functional Quads encoding / representations.
* Functional DOM REST HATEOAS Facades. Discovery.