* Ternary Semiotic Mappings
* Ternary Functional Encoding Functional Ternary Mappings Population:
* Ternary Mapping Form:
* Resource : (Resource, Resource);
* Graph Statement:(Node : S, Type : Ctx, Arc : P, Node : O);
* Parse Statements / Inputs.
* Mapping (A, B, C): Populate Contexts. Aggregate / Permutate Contexts recursively (data, schema, behavior layers roles).
* Hierarchy: Sign / Arc : Concept / Type : Object / Node;
* Monads / Lambdas / Functors: Sign, Concept, Object.
* Contextual Traversal Browsing of Resources / Kinds / State. Aggregation of Data, Schema, Behavior Augmentations: Aggregation, Activations, Alignments.
* Behavior:
* Behavior : Sign / Arc. Signs.
* Sign Mapping: (Concept Mapping, Object Mapping);
* Sign Mapping: Ternary Relations for Objects / Concepts Relation Mappings Contexts.
* Graph: Arcs / Signs.
* Schema:
* Schema: Concept / Type. Concepts.
* Concept Mapping: (Sign Mapping, Object Mapping);
* Concept Mapping: Ternary Relations for Object / Signs Relation Mappings Contexts.
* Graph: Concept / Types.
* Data:
* Data: Resource. Node / Object. Objects.
* Object Mapping: (Sign Mapping, Concept Mapping);
* Object Mapping: Ternary Relations for Resource / Resources Relation Mappings Contexts.
* Graph: Nodes / Resources.
* Mappings: CSPOs Resources Sign, Concept, Object Occurrences Mappings Contexts. Context Mappings Matching Sign, Concept, Object Contexts: Occurrences.
* Aggregated Mappings Browsing of Resources, Kinds / State Roles Relation. Functional Aggregation of Data, Schema, Behavior Augmentations: Aggregation, Activations, Alignments.
* Persist Quads. URNs Encoding allowing for HTTP REST HATEOAs / Functional Transforms / Traversal. Signatures (discovery / browse) contextual reactive event driven state APIs.
* Functional: Semiotic Ternary Mappings Layers Roles, Quads Parsing / Ternary Mappings Populations:
* Mappers:
* ObjectMapping (SignMapping : Occurrence, ConceptMapping : Context);
* ConceptMapping (SignMapping : Occurrence, ObjectMapping : Instance);
* SignMapping (ConceptMapping : Context, ObjectMapping : Instance);
* Mappings: Parse Quads. APIs.
* Resource : (URN, Referrer / Context);
* Resource : (Resource, Occurrences);
* Occurrences : Resource;
* Occurrence : Occurrences;
* Occurrences / Occurrence: Templated Resource URNs. Retrieveable Statements / Resources of Occurrence Contexts matching URN Resources "templates".
* Occurrences : Dereferenceable reified Resources URNs within a set of Occurrence Mappings. Resource Mappings / Graph Statements. Templated Resources.
* Occurrence : Dereferenceable Resource Mapping Item within a set of Occurrences. Context Occurrence Resource for further Next / Nil lists Occurrence Mappings. Mapping / Graph Items: Resource URNs plus Occurrence Context. Templated Resource.
* Templated Resource: URN Dataflow encoding of potential matching Resources predicates. Resolves to Occurrence / Occurrences in Occurrence Context Resource given Template URN encoding IDs values. Occurrences Template:
* CID:CID-CID:SID-CID:PID-CID:OID
* CID:CID: Are for Occurrence Mapping IDs (Sign, Concept, Object, whatsoever).
* The whole four quad segments are each for the corresponding Graph segments:
* (Node : Object, Type : Concept, Arc : Sign, Node : Object);
* The whole "address" admits query string manipulations and even give a different shape to resulting Statements. Besides "exact match" predicates, URN builder API (REST HATEOAS traversal) allows to craft meaningful Templates (Resource URNs, at last) reutilizable in other parts of the Models (applications) / Ontology (domains)
* Context: Role of Resource in Occurrence Statement.
* Dataflow.
* Facets: Resource for Role in Mappings.
* Role: Kinds / CSPOs.
* Ternary / Meta Models Layers Transforms / Mappings Relationships / Relations: Order, Context Comparisons, Translation, NER / CEP, etc.
* Matching. Augmentation.
* Encodings: SAILs:
* Ternary: Signs, Concepts, Values Aligned / Matching RDF I/O (Resources Graph Roles) SAIL.
* Ternary Sign Example: Sign Context, Occurrence Resource: Concept Concept Occurrence Resource: Object. Object Occurrence Resource: Sgn. Statements: Multiple Mappings Graph Reification.
* (Peter : Object, Loves : Sign, PeterLoves : Concept);
* (PeterLoves : Concept, Loved : Sign, Mary : Object);
* Model Layers: Meta Model, Discrete (Relationships), Activated Continuous (Dimensional) Models Statements Productions Encoding SAIL.
* Sets Model: Aggregated Statements, Kinds, Resource Sets Roles (Reified Kinds) Statements Rules Encoding SAIL.
* Dataflow SAILs: Layers Augmentations I/O (flows "dialog / prompts" fans in / out. Previous Layers feeds next until last feeding previous in a "bounce" manner previous Augmentation relation: Order, Conditionals, Wildcards, Placeholder "protocols".
* Ternary : Alignment / Matching
* Models: Activation
* Sets: Aggregation
* Ternary Semiotic Mappings
* Ternary Functional Encoding Functional Ternary Mappings Population:
* Ternary Mapping Form:
* Resource : (Resource, Resource);
* Graph Statement:(Node : S, Type : Ctx, Arc : P, Node : O);
* Parse Statements / Inputs.
* Mapping (A, B, C): Populate Contexts. Aggregate / Permutate Contexts recursively (data, schema, behavior layers roles).
* Hierarchy: Sign / Arc : Concept / Type : Object / Node;
* Monads / Lambdas / Functors: Sign, Concept, Object.
* Contextual Traversal Browsing of Resources / Kinds / State. Aggregation of Data, Schema, Behavior Augmentations: Aggregation, Activations, Alignments.
* Semiotics: DCI / MVC Augmentations / Dataflow:
* Upper Layers (hierarchical / contextual) domain driven entities / gestures / forms APIs.
* Ordered Graphs.
* Data:
* MVC: Model.
* DCI: Data.
* Graph: Nodes.
* Grammar: Terminal. Object.
* Functional: Monads wrapped Values.
* Augmentation: Aggregation (Data Matching).
* Component: Index. CDI CI Resolved bindings: Schema / Behavior Services Interfaces.
* Protocol: CRUD. Semantic Relations Queries. (e.g.: same as, data matching). Augmentations.
* Entities: Nodes. Data. Resources. Models / State. Arcs Recognition (Behavior). Objects From Schema. (Concept) to Behavior (Sign) Mappings. Model Resource Monads Values / Transforms.
* Schema:
* MVC: View.
* DCI: Context.
* Graph: Type / Context (Attributes).
* Grammar: Non Terminal. Concept.
* Functional: Monads wrapped Value Types.
* Augmentation: Activation (Schema Matching).
* Component: Registry. CDI DI Resolved bindings: Data / Behavior Services Interfaces.
* Protocol: Forms. Dataflow. Functional Flows: Initial / Contextual Form, Transforms (Graph Naming domain / range) arcs population (Index) Naming gives next populated state Form (Index retrieved state).
* Entities: Type / Context. Schema. Kinds / Roles. Node Recognition (Data). Concepts From Behavior (Sign) to Data (Object) Mappings. Domain Resource Monads Values / Transforms.
* Behavior:
* MVC: Controller.
* DCI: Interaction.
* Graph: Arcs.
* Grammar: Rules / Productions. Sign (recursion).
* Functional: Monads Transforms.
* Augmentation: Alignment (Behavior Matching).
* Component: Naming. CDI DC Resolved bindings: Data / Schema Services Interfaces.
* Protocol: IDs. Semantic Models (Graph, Dimensional, Discrete) Reactive Endpoints Encoding. Graph Name Flows Resolution / Entailments. Hierarchical Graphs Order Relations. Functional / Grammars Contextual (Transforms) state flows.
* Entities: Arcs. Behavior. Statements Kinds / NER Aggregation. Context Types / Relationships / Dimensions Recognition (Schema). Signs From Data (Object) to Schema (Concept) Mappings. Model / Domain Resource Monads Transforms Dataflow: Fixtures.
* Encoding: Semiotics Ternary URNs Mappings. Functional directed Ternary Mappings. Aggregation: Monads Traversal / Transforms Flows:
* Graph: (Node : Object, Type : Concept, Arc : Sign, Node : Object);
* Directional Mappings:
* Object : Node (Type : Concept, Arc : Sign); Data.
* Type : Concept (Arc : Sign, Node : Object); Schema.
* Arc : Sign (Node : Object, Type : Concept); Behavior.
* Graph Ternary Mappings URNs:
* ResourceID: (ResourceID, Occurrence ID);
* OccurrenceID : ResourceID;
* Graph URNs dereferenceable CSPOs Resources
* Funtional Ternary Semiotic Mappings URNs Naming Relations: Resource Encoding Roles / DCI Occurrences.
* DCI Models Encodings:
* (Occurrence, Context, Aggregated Type Attributes, Attributes Values : Occurrences);
* Graph Model: (Node : Data, Type : Context, Arc : Interaction, Node : Type Value);
* Dimensional Model: (Measure, Dimension, Unit, Value : Measure);
* Discrete Model: (Relation, Relationship, Kind, Entity : Relation);
* Labeled Property Graph: Arc : Node.
* Graph Model Dimensional / Discrete Models axes hierarchicaly ordered Measures / Relations relations Dataflow.
* Functional APIs:
* Model Monads: Models layers CSPOs / Semiotic Wrapper / Wrapped Types / Values. Layers Contexts. Augmentations.
* ResourceID:
* OccurrenceID:
* REST HATEOAS Dataflow:
* Graph URNs: Dereferenceable CSPOs / Ternary encoded Resources.
* Functional Dataflow: Monads / Transforms. Reactive REST HATEOAS. Dataflow Domain / Range Models / Domains Subscriptions.
* Domain Monads: DOM (Dynamic Object Model) Context Instances (Model Values) Entities.
* Functional Protocol: Dataflow / Parsing. Monads Wrapper / Wrapped Types / Values: Sign, Concept, Object Monads Contexts / Transforms Mappings. MVC / DCI Graph (Node, Type / Context, Arc, Node); Discrete / Continuous Models CSPOs Monads / Transforms. Model driven (Types Domain / Range ordered contexts) Monads / Transforms (Augmentations).
* Domain Monads: DOM (Dynamic Object Model) Context Instances (Model Values) Entities.
* Sample Statement:
* (NodeRID:OccurID, TypeRID:OccurID, ArcRID:OccurID, NodeRID:OccurID);
* Graph: Hierarchical Order / Dataflow.
* Graph: (Node, Type, Sign, Node);
* Events: (Relation. Relationship, Kind, Relation);
* Continuous: (Measure, Dimension, Unit, Measure);
* URNs: Resources / Occurrences. Encode CSPOs IDs: Ternary Dataflow Mappings. RIDs:OIDs: Roles IDs:
* CSPO Resource (Resource, Occurrence);
* Occurrence : Resource;
* Resolve entailments / inferences via Mappings Ternary Encoding Traversal ID Roles Augmentation / Aggregation / Activation / Alignments (Services).
* Ternary Mappings DCI Roles: Metaclass, Class, Instance, Occurrence, Context, Role.
* Co relations (Algebras / CoSQL / NoSQL / LInQ):
* Models. Layers: Augmentation steps reification. Layers Statements Roles.
* Base Model:
* (Context, Occurrence / Statement, Attribute / Kind, Value / Entity);
* Discrete Layers Model: (Relationships / Assertions). Continuos CEP: 0 Duration Measures / Axis Events Relations.
* (Relationship, Relation, Kind, Entity);
* Context, Roles, Causal, etc. Relations.
* Continuous Layers Model: (Dimensions / Measures).  Discrete CEP: Order / Containment Relations.
* (Dimension, Measure, Unit, Value);
* Distance, Equivalence, Transportation, etc. Relations.
* Functional DCI: Monads / Functors / Data.
* Wrapper Types (Data Roles). Encoding.
* Wrapped types (Data Values). Encoding.
* Functors: Dataflow Domain / Range specifications (Contexts). Encoding.
* Monads: Dataflow implementations (Interactions). Encoding.
* Functional Roles. Wrapped Models Resources Monads Wrappers Encoding (Dataflow Entities):
* DCI Data (ETL / Ontology Mappings). Layers: Augmentation steps reification. Layers Statements Roles.
* Class / Metaclass
* Role / Occurrence
* Value / Instance
* Attribute / Context
* Arcs: (Instance, Occurrence / Node Class, Attribute, Value); Data.
* Nodes: (Class, Metaclass, Context, Role); Contexts.
* Mappings: Models. Interactions.
* Layers Hierarchy: Role : Class; Value : Instance. Property Graphs.
* Layers:
* Arcs: (Role, Instance, Metaclass, Context); DCI Interactions.
* (Context, Role, Instance, Class);
* (Class, Context, Role, Instance);
* Nodes: (Instance, Class, Context, Role); DCI Contexts.
* Type: Class / Metaclass IDs
* Kind: Role / Occurrence IDs
* Node: Value / Instance IDs
* Arc: Attribute / Context IDs
* Layers:
* Arcs: (Arc, Node, Type, Kind); DCI Interactions.
* (Kind, Arc, Node, Type);
* (Type, Kind, Arc, Node);
* Nodes: (Node, Type, Kind, Arc); DCI Contexts.
* Models / Encoding:
* Layer Roles: (Context, Subject, Predicate, Object);
* Context: Kind Statements.
* Subject: Node Statements.
* Predicate: Arc Statements.
* Object: Node Statements.
* Type: Matching / Inferences.
* Encodings:
* Models: Quads / Property Graphs. DCI / Dataflow.
* Lists Model. Roles / Order.
* Hierarchical Graph Encoding.
* Sets.
* Functional DCI / Layers abstraction.
* Functional Parser. CUD. (data) of Grammar (Functional scheme) Dataflow Entities.
* Functional Parser. Grammar (Functional scheme) Dataflow Entities.
* Functional Parser. Parse (executions / behavior). Dataflow interactions instances.
* Codat: Dataflow / Protocol Prompts (run at).
* Protocol:
* Stateful I/O (ordered contexts). Reactive encoded Message driven gestures (CQRP).
* CDI / ESB Runtime. Backends. Connectors. CAM / CAN / DIDs.
* Dataflow HATEOAS.
* Augmentations / Alignments:
* Aggregation. Contexts / Occurrences.
* Activation. Roles / Types / Kinds.
* Alignment. Attributes / Values.
* Co relations (Algebras / CoSQL / NoSQL / LInQ):
* Models. Layers: Augmentation steps reification. Layers Statements Roles.
* Base Model:
* (Context, Occurrence / Statement, Attribute / Kind, Value / Entity);
* Discrete Layers Model: (Relationships / Assertions). Continuos CEP: 0 Duration Measures / Axis Events Relations.
* (Relationship, Relation, Kind, Entity);
* Context, Hierarchies, Roles, Causal, etc. Relationships / Relations.
* Continuous Layers Model: (Dimensions / Measures).  Discrete CEP: Order / Containment Relations.
* (Dimension, Measure, Unit, Value);
* Distance, Equivalence, Transportation, etc. Relations.
* Functional DCI: Monads / Functors / Data.
* Wrapper Types (Data Roles). Encoding.
* Wrapped types (Data Values). Encoding.
* Functors: Dataflow Domain / Range specifications (Contexts). Encoding.
* Monads: Dataflow implementations (Interactions). Encoding.
* Functional Roles. Wrapped Models Resources Monads Wrappers Encoding (Dataflow Entities):
* DCI Data (ETL / Ontology Mappings). Layers: Augmentation steps reification. Layer Statements Roles.
* Type: Class / Metaclass IDs
* Kind: Role / Occurrence IDs
* Node: Value / Instance IDs
* Arc: Attribute / Context IDs
* Arcs: (Instance, Occurrence / Node Class, Attribute, Value); Data.
* Nodes: (Class, Metaclass, Context, Role); Contexts.
* Mappings: Models. Interactions.
* Layers Hierarchy: Role : Class; Value : Instance. Property Graphs.
* Layers:
* Arcs: (Role, Instance, Metaclass, Context); DCI Interactions.
* (Context, Role, Instance, Class);
* (Class, Context, Role, Instance);
* Nodes: (Instance, Class, Context, Role); DCI Contexts.
* Layers:
* Arcs: (Arc, Node, Type, Kind); DCI Interactions.
* (Kind, Arc, Node, Type);
* (Type, Kind, Arc, Node);
* Nodes: (Node, Type, Kind, Arc); DCI Contexts.
* Mappings: Models. Interactions.
* Layers Hierarchy: Node : Arc;. Property Graphs.
* Models / Encoding:
* Layer Roles: (Context, Subject, Predicate, Object);
* Context: Kind Statements.
* Subject: Node Statements.
* Predicate: Arc Statements.
* Object: Node Statements.
* Type: Matching / Inferences.
* Encodings:
* Models: Quads / Property Graphs. DCI / Dataflow.
* Lists Model. Roles / Order.
* Hierarchical Graph Encoding.
* Sets.
* Functional DCI / Layers abstraction.
* Functional Parser. CUD. (data) of Grammar (Functional scheme) Dataflow Entities.
* Functional Parser. Grammar (Functional scheme) Dataflow Entities.
* Functional Parser. Parse (executions / behavior). Dataflow interactions instances.
* Codat: Dataflow / Protocol Prompts (run at).
* Protocol:
* Stateful I/O (ordered contexts). Reactive encoded Message driven gestures (CQRP).
* CDI / ESB Runtime. Backends. Connectors. CAM / CAN / DIDs.
* Dataflow HATEOAS.
* Augmentations / Alignments:
* Aggregation. Contexts / Occurrences.
* Activation. Roles / Types / Kinds.
* Alignment. Attributes / Values.
* Ontologies: Gods / Products / Needs. Showcase. Transaction. Learning / Inference / Entailments, Workflows. Domains Alignments. Campaigns. Support Systems / Services Protocols Alignment Ontologies.
* Funtional Semiotic Mappings. Relations: Resouce Encoding Roles Roles / Occurrences:
* Metaclass
* Class
* Instance
* Context
* Occurrence
* Role
* Models. Layers: Augmentation steps reification. Layers Statements Roles.
* Base Model:
* (Context, Occurrence / Statement, Attribute / Kind, Value / Entity);
* Discrete Layers Model: (Relationships / Assertions). Continuos CEP: 0 Duration Measures / Axis Events Relations.
* (Relationship, Relation, Kind, Entity);
* Context, Hierarchies, Roles, Causal, etc. Relationships / Relations.
* Continuous Layers Model: (Dimensions / Measures).  Discrete CEP: Order / Containment Relations.
* (Dimension, Measure, Unit, Value);
* Distance, Equivalence, Transportation, etc. Relations.
* Functional DCI: Monads / Functors / Data.
* Wrapper Types (Data Roles). Encoding.
* Wrapped types (Data Values). Encoding.
* Functors: Dataflow Domain / Range specifications (Contexts). Encoding.
* Monads: Dataflow implementations (Interactions). Encoding.
* Functional Roles. Wrapped Models Resources Monads Wrappers Encoding (Dataflow Entities):
* DCI Data (ETL / Ontology Mappings). Layers: Augmentation steps reification. Layer Statements Roles.
* Type: Class / Metaclass IDs
* Kind: Role / Occurrence IDs
* Node: Value / Instance IDs
* Arc: Attribute / Context IDs
* Arcs: (Instance, Occurrence / Node Class, Attribute, Value); Data.
* Nodes: (Class, Metaclass, Context, Role); Contexts.
* Mappings: Models. Interactions.
* Layers Hierarchy: Role : Class; Value : Instance. Property Graphs.
* Layers:
* Arcs: (Role, Instance, Metaclass, Context); DCI Interactions.
* (Context, Role, Instance, Class);
* (Class, Context, Role, Instance);
* Nodes: (Instance, Class, Context, Role); DCI Contexts.
* Layers:
* Arcs: (Arc, Node, Type, Kind); DCI Interactions.
* (Kind, Arc, Node, Type);
* (Type, Kind, Arc, Node);
* Nodes: (Node, Type, Kind, Arc); DCI Contexts.
* Mappings: Models. Interactions.
* Layers Hierarchy: Node : Arc;. Property Graphs.
* Models / Encoding:
* Layer Roles: (Context, Subject, Predicate, Object);
* Context: Kind Statements.
* Subject: Node Statements.
* Predicate: Arc Statements.
* Object: Node Statements.
* Type: Matching / Inferences.
* Augmentations / Alignments:
* Aggregation. Contexts / Occurrences.
* Activation. Roles / Types / Kinds.
* Alignment. Arcs. Attributes / Values.