* FCA Implication confidence / support.
* ResourceOccurrences (CSPOs, Kinds, Statements) / Resources: IRI, (Contexts, Kind)\* Ocurrences.
* Rules Alignment: RuleConcepts / RuleImplications. Aggregated from CSPO Occurrences, Kind Relationships, Statements / KindStatements Relationships. Rule / Implication: (Context, LHS, Concept, RHS); Populate Rules with Upper Alignment knowledge and aligned Model inferred knowledge.
* Rule / Implication: (Context, LHS, Concept, RHS) Inference: Kinds Instance, Attribute, Value CSPO Kinds Contexts aggregation / subsumption (domain / range flow). Matching: CSPO Kind Attributes matching those of Rules / Concepts.
* Relationship Statements: Rules (LHS, RHS in a Context by a Concept Property) stated via Upper Aligned types, instances, attributes, values Statements, Kinds, CSPOs and Resources (Concepts / Implications).
* Relationships / Rules Model. Model Aggregation / Inference.
* Implication (instances): Statements, Kinds, Resources. Parse instances as new Rules / Concepts.
* Model (Upper Aligned CSPOs):
* Rule / Implication: (Context, LHS, Concept, RHS);
* Context, Concept : Rule / Implication.
* LHS, RHS : ResourceOccurrence (Statement / Kind / CSPO) wrapper Monad.
* Kinds : Function::apply(Kind arg) : Kind (resolve instances, attributes, values occurrences).
* ResourceOccurrence wrapping Monad (Resource, Context, Kind), Kind Functions (domain / range, subsumption).
* Resolve Context RHS Kind / CSPO Resources.
* Apply Rule LHS Kind against Context RHS Resources output.
* Apply Concept LHS to previous Kind Resources output.
* Apply Rule RHS to Concept RHS Kind Resources output.
* (Context, LHS, Concept, RHS); Upper / Aligned Primitives: (LHS, RHS). Primitive Contexts, Concepts.
* Rule Aggregation: Each Statement is itself a Rule stating a single fact building a Concept by means of its Kinds relationships / CSPO inter Statements occurrences.
* Initial Rules / Implications: initial Statements, KindStatements. Aggregate / Align Statements CSPO Kinds / KindStatements Rules / Implications by their attribute / value CSPO types Kinds.
* Rule: (Kind) Statement / Kind / Resource matching LHS (Upper onto matching aligned), (Upper / Model inferred) Concept in Context, (Kind) Statement, Kind, Resource RHS.
* State: Type according property values / property values relationships. Alignment, Augmentation: shapes (Upper OWL, SHACL, ShEx). Metakinds (person / adult: age value gt 21; brother / uncle: brother :hasChild). Order relationships (Statements). Activation (Context inter-resource relationships). Reification: express inter (Kind) Statements relationships. Statements ResourceOccurrence (Kinds / CSPOs) of Concepts Resources. Relationship Statements (upper / shapes).
* Upper Ontology / Matching: State implies Event / Relationship / Type: maritalStatus, uncle. Employee isA Worker. State facts by means of Upper types, instances, properties, values, relationships (RelationshipStatements: Statements about Statements, Resources, Kinds concepts).
* RelationshipStatements (Kinds / CSPO, Relationships (reified Statements). Aggregate inferred rules / logic statements (Resolvers / Notation3). Upper ontology types, instances, properties, values, relationships (sameAs).
* Upper Ontology Relationships: property values according type. before, during, after. Greater than, less than, equal. Event. Cause / Effect (state flow). Contains (Relationship, Place, Duration). Relationship kinds. N-ary Relationships (roles, actors templates).
* Augmentation / Alignment: Attribute / Links prediction. Upper Ontology alignment. Ontology Matching. Predicted Attributes Values.
* Activation: Kinds order (in Relationship Contexts).
* Order Statements by Kind hierarchies, previous / next common Attributes values.
* Label (add to Resource) Resolvers. RDFS#label NLP NER.
* Upper Ontology: ISO Topic Maps (TMDM, TMRM), ISO 15926 (OWL Templates).
* Statements / Kinds: order. Domain / range dataflows. Alignment, link prediction (between Kinds, KindStatements FCA).
* Activation: Kinds Functions (order: domain / range). Apply (map) adds (transition fits by schema) / query for Attributes / Values to Resource Kind Monad occurrence.
* Activation: Order Statements / Kinds transitions. Single, Married. Married, Divorced. Temporal order (octal comparison values encoding). Previous(Current / Next), Current(Previous / Next), Next(Current / Previous) Kinds.
* Inferred Attributes: Value Regression / Prediction (occurrences model). Scaling, MultiValuedContext.
* Inferred Attributes: Values prompts / placeholders (Resolvers).
* CSPO Kinds: Parameterized Instances, Attributes, Values.
* Kinds Impl: get inst (instances, instance), attrs(inst), vals(inst, attr) CSPOs.
* Kinds get Instance, Attribute, Value CSPOs. Fix / Move to CSPO Kinds.
* Input Statements: (Class, Instance, Attribute, Value). From RDF4J Model / Query.
* Query Repository rdfs:type value: Context, SPO.
* Kind Labels: from string similarity Subjects / Predicates NER.
* Reduce Context (FCA).
* Kinds hierarchies: Concepts subsumption.
* Activation: SubjectKinds / ObjectKinds: DCI Roles. PropertyKinds: Roles Behavior (Kinds Functions). KindStatements: DCI Contexts. Statements: DCI Interactions. CSPOs: Actor Occurrences.
* Alignment / Activation: Materialize Statements from KindStatements Kinds instances, attributes, values. FCA Objects / Attributes: Kinds.
* Activation: Rest (JAF), Restful Objects, HAL, Spring REST. HATEOAS.
* Alignment: Set KindStatement Kinds Kind.
* Aggregation: returns KindStatements. Kinds.getCSPOs.
* Resource: getOccurences.
* Scaled Attributes: Values.
* Kinds / Statements occurrences Resources, Kinds (roles), Statements (RDF star, reification).
* CSPO Statements.
* KindStatements
* RelationshipStatements (Kinds / CSPO, Relationships (reified Statements).
* KindStatement: schema Statements.
* Kinds Statements implements Kinds interfaces, Statement extends ResourceOccurrenceImpl (Context Statement Kinds: roles / metaclass: domain / range). CSPO Statements: Kind Statement Instances (Kind Statement getStatements).
* Reification Statements: example: SubjectKind implements Subject. Subjects SubjectKinds SubjectKindStatements.
* Kind Statements (Aggregated): SubjectKindStatement extends Statement. C: Super Kind, S: Kind, P: PK (domain), O: OK (range), From merged Occurrence Context Statement Kinds. Kind Statement Kinds: role / metaclass.
* CSPOs: Resource IRI / PrimeID. Merged Kinds: Instance, Attribute, Value Context Lattice products.
* Resource: Function and Monad wrapped value. Domain / range state / flow context / interactions order. Relationships. Activation.
* Statements URN: lookUp by CSPO hashCode (Map / Embedding).
* Statement IRI: Reification, RDF-Star (URN Encoded).
* Resource PrimeID, Resources URN Encoding.
* Kinds Aggregation: Merged Resource IRI (Instance, Attribute, Value URN). Prime factors ID. FCA4J.
* FCA Contexts:
* getContextAggregatedKinds
* getSubjectAggregatedKinds
* getPropertyAggregatedKinds
* getObjectAggregatedKinds
* FCA. Primes Embeddings. Property contexts, Subject objects, Objects attributes. Aggregate by Property contexts. Lattice Graph (Alignment).
* Serialize Aggregation, Alignment, Activation into RDF / RDFS / OWL / JSON-LD: Instantiate Model via RDF4J APIs. (Align upper ontologies / [schema.org](http://schema.org)).
* JSON-LD Serialization (ObjectMapper). Types documents: Kinds, Resources. AggregationService performAggregation.
* Convert any input into triples graph (Statements). Alignment: Graph Deep Learning). DOM Mappings. UIMA / Tika / any23 / D2RQ.
* Kinds: Initial Kind Resource: Kind Instance Resource. Aggregate: Instances Context (Attrs / Values).
* Target: Generate typed RDF RDFS / OWL (Kinds). Sets Aggregation.
* Target: Discover Links / Relationships. DOM Augmentation.
* Target: Discover Data Contexts / Interactions. DCI Activation.
* Kinds : Statement (CK Kind IRi)
* Kinds interface / impls: get / set Inst, Attr, Val (Statement ctx).
* getSubjectAggregatedKind()
* Resource / ResourceImpl occurrences / hashCode.
* CSPOs, Kinds, Statements: IRIs Occurrences roles. IRI in CSPOs, CSPOs in Kinds / Statements, Kinds in Statements. Resource get occurrences (Subjects, etc. streams / filter). Resource super type (IRI / Occurrences) CSPO / Kinds / Statement parameterized (Statement Resource<CSPOs, Kinds>).
* Statements: Model / Model Statements wrapper. Streams (load statements, aggregate kinds, populate CSPOs, filter / occurrences: CSPO / Kinds Statements). Statements (filter / occurrences set streams, Statement instances).
* Todos los atributos de un Subject son un Kind: CK (Kind Statements Stream). Kinds (filter / occurrences sets streams, Kind instances).
* Merge Kind Statements. Kind IRI: Statement Contexts.
* Populate CSPOs. Filter Statements Stream. Subjects, etc. (filter / occurrences set streams, Subject, etc. instances).
* Quads (Resource) : Type, Instance, Attribute, Value (parameterized). Interfaces.
* Resource : IRI, Occurrences (CSPOs, Kinds, Statements). Implementations.
* Context
* Subject (SK, IRI, Ps, Os)
* Property
* Object
* SubjectKind (Super SK, SK, Ps, Os) : P / O
* PropertyKind : S / O
* ObjectKind : P / S
* ContextKind : (S, P, O)
* Statement (CSPO, CK:SK:PK:OK Combinations, SK en CK: Ps / Os).
* Functions (Transforms) : Reactive.
* Getters: Type, Instances, Attributes, Values
* Getters CSPOs, Kinds, Statements Occurrences
* Alignment: super Kind infer sub Kind (superset) Attributes (Aggregation) / Values (Regression).
* Aggregation: Kinds (IRIs) / Instances (Statement Occurrences) Attributes / Values. Sets Model. Classification (no labels)
* Alignment: Types / Instances Relationships / Links. Kinds / Attributes. DOM Model (DAOs).
* Activation: Roles / State in Contexts / Interactions Relationships. DCI Model. Clustering.
* FCA Embeddings.
* Statements / Kind Statements : IRI (toString / hashCode). Reification.
* D2RQ Sample Dataset (Cinema).
* Aggregation: Materialize Kind Statements (RDFS / OWL).
* Debug Aggregation results.
* Retrieveable IRIs : JSON-LD. Alignment: [schema.org](http://schema.org).
* Alignment: sameAs.
* Context Kind: Context, Predicate, Object.
* Kind (role) in Occurrences / Context Kind Statements. DCI Context, DOM Alignment.
* SK: Subject, Predicate, Object.
* Debug: Aggregation. Sample relational dataset (D2RQ) FKs (inferred / Alignment). Browsable Aggregation output format: Statements, Materialized Kind Statements: RDFS / OWL (Semantic Web / SPARQL Browser).
* Reactive Functional / Async:
* Flat map each 3 aggregations from initial flows, each return next aggregation input. Flat map 3 aggregation groups.
* FKs. Objects / Subjects referred by IRI, Aggregated end types (Kinds / Alignment), Relationship Kind (PredicateKind). Star Schema (Dimensions, Facts).
* IRIStatementOccurrence / Kind getStatementOccurrences.
* Alignment: query / views / rules parameters: add OWL Templates to Repository Model. DOM DAOs.
* Materialize inferred Statements (reified Kind Statements) in Repository temporal Model. Alignment: SPARQL query templates / views / rules. RDF4JTemplate: DOM DAOs. Services.
* Materialize RDF rdf:class (Kinds), rdf:type, etc. Inferred attributes. Alignment queries.
* Materialize: Statements: Reification / RDF\*. Statement : rdfs:Statement. IRIStatementOccurrence. Statement / Kind Statement IRI.
* Handle Literals: Literals, Map<hashCode, Literal>. IRI: Literal type + hashCode.
* toString: Complete Entities / FKs JSON. Messaging: parse Resources toString. Autowired Custom ObjectMapper.
* Convert Aggregation to Functional / Reactive. Integrate with RDF4J.
* Aggregation. Learn Types.
* Alignment. Learn Relationships.
* Activation. Learn Interactions.
* Alignment: Functional / Reactive APIs. DOM / DAOs. Embeddings.
* Alignment Service: align(), get Types, Entities, Attributes, Values. RDF4JTemplate DAOs.
* Resource (HasIRI) interface. Resource::apply(Resource res) : Resources; Dynamic Functions.
* Map / flatMap Model domain (Static Functions).
* Zip T1, T2 (Resources product): apply T2 to T1 (Dynamic Functions) : Resources.
* Alignment: Events, Order relationship. Inferred.
* Update RDF4J with Alignment inferred Relationships (query / rules SPARQL).
* Emit inferred Statements.
* TODO:
* Kind has Attributes.
* Attributes has Kind according Occurrence (Instance).
* Attribute has Value according Instance.
* Values has Attribute according Occurrence (Kind).
* Replace placeholder IRIs with primes product URNs (Embeddings).
* Kind, Instance, Attribute, Value: IRIs / Embeddings.
* Primes.common(BigInteger, BigInteger).
* Primes.contains(BigInteger, BigInteger).
* Kinds Naming: Context IRI. KindStatements. Contexts Label.
* Aggregation: order. Kinds sub / super Kinds relationship encoded in Statement Context IRI.
* Kind reification: S : (ParentSK, SK, P, O);
* Alignment: Links Attributes / Values inference / materialization.
* Alignment: Theory. Order. State. Rules. Events. Relations. Uncle : (aFather, fatherBrother)
* Alignment: Attribute / Value clustering determines meta-Kind (age > 21 : Person age Adult Adult Person); (birthPlace BsAs, Person nationality Argentina, Argentinian Person).
* Alignment: Upper Ontology. Mappings.
* Alignment: Ontology Matching.
* Activation (DDD DOM / DCI). REST Dialog front end.
* Activation Service: activate(), get Interactions, Roles, Actors. RDF4JTemplate DAOs.
* Embeddings calculation (FCA / primes. Contexts / Attributes / Objects / Concepts.
* FCA Contexts: Axis, Objects, Attributes, Concepts.
* Restful DOM / OGM (DCI, CDI)
* Activation: Gestures (available Transforms).
* HATEOAS REST (HAL) Architecture. Components:
* Alignment / Activation DOM (RDF4JTemplate DAOs):
* Entity
* Name
* Type
* Value
* atributes : Set<Entity>
* Type : Entity
* Name
* players : Set<Entity>
* members : Set<Type>
* Role : Type (Wrapper)
* Type
* Declarative Behavior / Relations?
* Context : Role (Composite)
* Entity
* Type
* Role
* Declarative Behavior / Relations?
* Interaction
* bindings : Set<Context>
* Model:
* Statement : Resource
* Context : Context
* Subject : Subject
* Predicate : Predicate
* Object : Object
* Kind : Statement
* Kind<INST super Resource, ATTR super Resource, VAL super Resource>
* Context : Kind<> (this)
* Subject : INST
* Predicate : ATTR
* Object : VAL
* List<Kind<>> : statements
* Context : Resource (Wrapper)
* Statement : context
* ContextKind : role
* player : Resource
* ContextKind : Context, Kind
* extends Kind<Context, Subject, Predicate>
* Subject : Resource (Wrapper)
* Statement : context
* SubjectKind : role
* player : Resource
* SubjectKind : Subject, Kind
* extends Kind<Subject, Predicate, Object>
* Predicate : Resource (Wrapper)
* Statement : context
* PredicateKind : role
* player : Resource
* PredicateKind : Predicate, Kind
* extends Kind<Predicate, Subject, Object>
* Object : Resource (Wrapper)
* Statement : context
* ObjectKind : role
* player : Resource
* ObjectKind : Object, Kind
* extends Kind<Object, Subject, Predicate>