Distributed Integration and Consistency for Knowledge Semantic Interoperability

Outline (drafts)

Purpose (integration, augmentation, extension)

Description

Analysis (mission)

Design (vision)

Deployment Use Case: Goals App Solution

Features

Connectors

Clients

Adapters

Declarative (model / domain driven):

Domains alignment / matching: aggregations / learning (data, schema, behavior information / knowledge augmentation)

Sources augmentation (reactive synchronisation)

Shared data consistency: Governance over (distributed) MDM, BPM, Business Rules. Distributed inference repository

Problem spaces facades: inter domain / inter apps unified dashboards. Goals gestures / dialogs (Form / Flows) translation.

Domains extension (Protocols / APIs):

Microservices Hub API. Declarative (model described) services APIs / Protocols (Activation integrable)

Protocol: hierarchical contexts selection / edition of role values (reified possible contexts. Services Activation context state protocol (Activation grammars)

Distributed Persistence:

Event sourcing / Saga pattern: distributed addressables queue. Reconciliation. Trust. Transactions: Request IDs.

Addressable Interactions. CQRS. Reified interactions / gestures. Logs. Reified (resource) Message Activation (dataflow). Forms / Flows

Implementation:

Model: RDF CSPO Quads store. RDFS / OWL inference.

Primitives / meta / upper resources. Grammars: patterns / signatures. Context Kinds (Message / Augmentation)

Kinds: hierarchies. Semiotic / Sets DM / RM encoding.

Layers. Streams. Events. Augmentation.

Resource Monad (REST). Addressable Interface. DOM (roles, rels, members, etc.) Functors.

Form / Flow Facades

Message Functor. LHS: domain: source / RHS: range: dest. Augmentation / Event pattern declaration. Mapping encoding. To do.

Model Context hierarchy (DOM). Layers CSPOs classes DOM: type / context, occurrence, attribute, value, prev, next, parent, children, etc. members. Functors.

Message: Augmentation (event). declaration.

Core: primitive / meta / upper resources. Reified layers contexts. Order (dataflow) encoding.

Core: Augmentation. Aggregation (pragmatics, roles). Alignment (semantics, rels). Activation (levels: schema, grammar, syntax of roles / rels).

Core Contexts. Layers. Augmentations.

Domains (Context DOM instances).

Dataflow: Events I/O. Message augmentations. Connector / Client / Adapter.

Message Mapping Augmentation perform. XML / XSL over RDF (cons) lists. DOM members resolution (functors) / embedding.

Aggregation / Matching (Model / Entity, Behavior / Measure). OntResource embedding (contextual functor).

Events I/O "fans in/out" model layers performing Augmentation Activation.

Graph key / value (properties) encoding. Map Reduce event flow tasks.

Deployment:

Container: Forms / Flow DCI / MVC Facade APIs. Resource (REST) Monad and Context Functors (reified Message resources). Protocol (Connector / Client Adapters).

Low level Resource API.

DOM / OGM rendering / activation APIs.

Model Context DOM members (layers): order (prev, next, parents, children) in axis. Metaclass, class, instance, occurrence (roles / rels in contexts). Functors / DOM events API: request reified Message resource in "possible" (domain, mapping, range) context. Bindings (subscriptions).

Items:

MDM. Provenance. Versioning. Dimensional context values / queries. Model Facets APIs: Functional, Semiotic, Dimensional Dataflow contexts / order / roles HATEOAS APIs workflows. Protocol: Dialog. Browse / analyze / transform "activations" (REST / JAF) according Facets. Clients / Connectors.

Contents / Features (Mision / Vision). Distributed consistent Knowledge Applications. Trust. Consistency. Event sourcing. Inferencing (of distributed state). Reconciliation.

Certify distributed Entity / Subject Identity / State (in roles / dimensional points). Class / instance alignment (matching).

Integration: Augment sources / back ends. Model I/O materialized in source (plugged) application / services back ends.

Integration: Extension. Extended functionalities data / schema / behavior exposed as services external to source (plugged) applications. Sync (Augment). Declaratively stated via Model descriptions. Discoverable, browseable (HAL / REST).

The idea of the project is to "augment" an ESB for EAI platform and to enable it allowing it to make "inferences" regarding which routes to use, "discovering" sources / destinations of an event message(s) which then it transforms / enriches according destination "semantics" and format(s).

This featuring the exposure of a generic facade which allows to see in an "homologated" view the applications or services and their data, schema and behavior (actions) that could be integrated into the tool.

Different integrated applications are enriched with this facade and with the events that, given the inferred routes and transformations, augments theirs data, schema and behaviors, invoking activities corresponding to each destiny semantics.

Resource monad of Model(s) layers Contexts: Transform, Kind, Entity, Behavior, etc. Model(s) Context hierarchy classes (model) instances (domain). Contexts, Occurrences, Attributes, Values : CSPO Functors (eval in Resource contexts). Functor declarations (model augmentations / domain dataflows encoding).

Monads / Functors examples (order: comparables / upper / dimensional ontologies):

Kind<SubjectClass<Resource>, Set<PredicateClass<Resource>>>, others. Example: Subject / Predicate(s). (quads prev / next relation).

Metaclass<ObjectClass<Resource>, Set<OccurrenceClass<Resource>>>, others. Example: Object / Statement(s). (layers parent / child occurrences relation).

Class<ObjectClass<Resource>, Set<ContextClass<Resource>>>, others. Example: Resource / Kind(s) (class definition relations: extension / intension, layers parent occurrences prev relation).

Backend: RDF(S), OWL, Resource Services (Naming, Index Registry), ISO, Shapes, IDs matching. DIDs Backend.

Runtime: Objects, Events. Distributed Runtime Functional (Dataflow) reactive Resource objects / models (forms / flows).

Deployment: Spring / Vertx / others (Jersey / CDI: Resources Meta Model / Domain APIs). HATEOAS / HAL APIs (Encoding / Dialog Endpoint Protocol). Environment / Container: Messaging / Services. Models I/O.

Integration Connectors: Runtime Model embedded Resources. Sources as reactive stream objects. API.

Integration Clients: Runtime Model embedded Resources. Services as reactive stream objects. API.

Integrations (augment / extend): RDBMS (R2QL / R2ML) / Rules / BPM, KIE / Teiid / Metamodel / Olingo / Solid / ERP / CMS / Feeds (Hypermedia Resources / Apache Stanbol) / Browser / Messaging (JMS overlay, Apache Camel / ActiveMQ: ServiceMix Connectors / Clients) / ML, Big Data (Apache Spark).

Dialog: Encoding / Protocol. Activation, Location, Context (Hypermedia type / context interfaces). Augmentations.

Encoding: XML / XSL / XPath / XLink / XForm / XPointer / XQuery (RDFS / OWL / ISO DM / RM Forms / Functors / DOM). Endpoints (streams / signatures): documents (messages) / resources (paths).

Comparable: parent / child relations (SortedSet). Model sets / dimensional alignments.

Models: Augmentation, Kind (Metamodel), Entity, Class, Behavior (Functional), Dimensional, Semiotic (Matching / Alignment / Upper).

Runtime. Bus. Signatures bindings (reactive), content type / domain context interactions data transforms (roles).

Design: Augmentation (Aggregation, Alignment, Activation Functors), Domain Functors. Encoding. Design Functors Dataflows.

CQRS. Event Sourcing. Functor Commands.

Protocol / Encoding: hierarchical contexts dialog (runat) prompt / pick / select (roles).

Endpoints: Model (classes) objects / resources, Domain (instances) objects / resources

Apache Isis (DDD / Actors). Connector / Client / Runtime (Hypermedia / Facade / Dialog Protocol: services / actions). Backend: Augmented Services / Persistence. Representations (resource: types / activation). Domain, Services, View Models.

JBoss Teiid / Apache Metamodel: Clients / Connectors / Backend / Runtime (Translators).

CMS / Wiki API / WebDAV. Apache Stanbol. DOM / REST / JAF.

StratML Client / Connector (Goods, Needs, Products, Goals, Purposes. Exchange ontologies).

Ontology Matching: Dimensional metaclass / class / instance attributes relationships in axes / contexts / roles evaluations. Resolve equivalence of dimensions, units, measures, values via aggregation of value occurrences. Resolve value kinds by occurrences of Semiotic (grammar) model. Contexts (CSPOs) Attribute / Value "clustering". Sets specifications. Dimensional order relations.

Distributed Inference (dimensional / matching / predictions). Distributed consistency. Event sourcing (Dialog, CQRS Protocol). HATEOAS distributed / lazy reconciliation: hashing of source / destination (hierarchical contextual model quads hashing) state in interactions tokens. Prompts. Distributed Alignment (until reach of goal synchronization states). Distributed state / IDs.

Comparable: parent / child relations (SortedSet). Model sets / dimensional axes alignments.

Semiotic: syntax, semantics (grammar), pragmatics. Verbs: action, passion, state (for / due to action / passion). Models. Dimensional (axis / order) relations. Alignments. (Sorted) Set relations: hierarchical encoding. Verb roles (CSPO / Kinds sets).

Example aggregation: candy (type), red (color attribute), strawberry (flavor attribute).

Encode IDs: Context Kind, upper (meta) Resources (levels / layers). Resource contents / contexts (identify by occurrences in roles in other contexts, Meta Resources, layers class, metaclass, instance). Compose IDs (hierarchical graph properties encoded string) from outer to inner resources (Context, Kind, Occurrence, Role, Resource). "Operable" IDs (ClassIDs / InstanceIDs: Meta Model reifications / occurrences).

Message parsing (Template, Message Context) matches Form / Flow Augmentation Mappings signatures.

(Template, Context, Attribute, Value); Value as Context: hierarchical models. Same attributes: types / collections.

Semiotic (encodings):

(Context, Sign, Concept, Object);

Object as Sign: Object properties (Concepts).

Semiotic encoding: encode Meta Model (syntax facet) / Functional (Meta Model encoded pragmatic facet) / Dimensional (Meta Model encoded semantic facet) data / reference model (by contexts / upper alignments / aggregation).

Facets contexts semiotic encoding. Facets IO (events) by semiotic encoding of facets input layers.

(Context, Sign, Concept, Object);

Roles encoding: Object as Sign (properties), Sign as Object (types / roles), Concept as Sign / Object / Context, etc.

Grammars encoding: express models contexts layers and augmentation templates in input contexts.

Meta Model:

(Transform, ID, ID, ID);

(Mapping, Transform, ID, ID);

(Template, Mapping, Transform, ID);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping);

(Role, Resource, Augmentation, Template);

(Statement, Role, Resource, Augmentation);

(Model, Statement, Role, Resource); Resource Occurrence in Model (Kind? Kind Role type, Kind hierarchies.)

Functional Facet:

Occurrence

Entity

Kind

Class

Flow

Behavior

Dimensional Facet:

(Value, Previous, Distance, Next);

(Measure, Value, Previous, Distance);

(Unit, Measure, Value, Previous);

(Dimension, Unit, Measure, Value);

(Concept, Dimension, Unit, Measure);

DCI / DOM: Subject, Context, Occurrences, Roles, Attributes, Values, Activation.

Semiotic, Meta Model, Dimensional, Functional DOM. DOM Contexts. DOM Functors. Resources Contexts (CSPO Monads: encoding / addressing).

Activable DOM Contexts: DOM views (object mappings, etc.).

Map Reduce encoding inputs, grammar templates context mappings. Emit Semiotic reference / data model parent / child properties encoding (Context / SPOs, Context:Subject / POs, etc.).

Model:

Integration: Purpose (description, mision, vision).

Task list focus for each item.

Analysis (mision).

Design (vision).

Containers / Components / Addressing / Routing / Protocols / Interfaces / Services (Message IO data / Backend Models schema / Behavior encoding).

Deployments (use case) Goals app (tasks, exchanges, etc. over integrated backends: "wizards").

Model / Semiotic reification.

Facets / Levels / Shapes: Aligned entities / values occurring in aligned models / dimensions.

(Context, Sign, Concept, Object);

Resource Monad / Message Functor (Contexts hierarchy parent). Resource<Context>. Events bindings. Message / Augmentation event declarations / instances.

Resource

(Message, LHS, Cond, RHS);

(Context, Sign : Message, Concept : OntResource, Object : ID Resource);

(Transform, Context, Message, Concept);

(Mapping, Transform, Context, Message);

(Template, Mapping, Transform, Context);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping);

(Role, Resource, Augmentation, Template);

(Statement, Role, Resource, Augmentation); Augmentation of which Statement is result of.

(Model, Statement, Role, Resource);

(Entity, Model, Statement, Role); Model (Backends) aligned entities.

(Kind, Entity, Model, Statement);

(Class, Kind, Entity, Model);

(Flow, Class, Kind, Entity);

(Behavior, Flow, Class, Kind);

(Measure, Behavior, Flow, Class);

(Value, Measure, Behavior, Flow);

(Unit, Value, Measure, Behavior);

(Dimension, Unit, Value, Measure);

Functional: Entities in different Models in Kind, Class, Flow, Behavior.

Dimensional: Measures in different Dimensions in Units, Values.

Value, Previous, Distance / Event, Next. Order (in axis):

Siblings: previous / next (Semiotic containment relationship / roles). To do.

Hierarchies: parent / children (Semiotic containment relationship / roles). To do.

Semiotic model: reification / primitives / upper ontology (meta model, inferred / aggregated). Levels: syntax / grammars, semantics, pragmatics. To do.

Model (meta / upper resources) / DOM (domain instances) Augmentation Functors:

Aggregation: Create. Context occurrences. Materialise into next layer(s) streams. Context occurrences statement(s) (children: next layers). To do.

Alignment: Update / Retrieve. Augment attributes / values. Materialise into next context augmented statement streams (next context occurrence attributes / values, layer siblings). To do.

Activation: type / role in contexts. Parent (occurrence Context) occurrence Kind for Context object (parents: previous layers). To do.

Activation: type / role in contexts. Parent layer (occurrence context layer) CSPOs Kind(s). Materialise into dataflow Message(s) streams: (CK, SK, PK, OK). (parents: previous layers). To do.

Example:

Entity in (Model, Statement, Role, Resource).

Kind in (Entity, Model, Statement, Role).

Aggregation / Alignment Messages from primitives / meta / upper / domains resources Activation.

CSPO: Context (intension), Object (extension). SortedSet comparison hierarchies (encode order in semiotic DM / RM, octal comparison values).

Message: Augmentation (materialised Resource set) specification (functor). Event declaration (I/O patterns bindings). Context layers monad publishes / subscribes (dataflow bindings pipelines).

Addressing (hypermedia):

Content Type: Metaclass

Resource: Class

Representation: Instance

Address: Occurrence / Response (Materialised Message Augmentation)

Request: Message (state flow). Context DOM event API: Request Message Resource in possible domain / range / mapping contexts.

URIs: Contexts (data), Types (reified schema), Service (behavior). Class, instance, occurrence links, paths, pointers (annotations / transforms / embeddings / calculated / descriptions).

Persistence: distributed event queue (saga) of distributed addresses (occurrences metaclass, class, instance resolution). Composite occurrences (links / rels / roles).

Purpose / content type driven (state / rendering / roles / links /contexts / data / flows / attributes) declarative hypermedia activation application. Model, Application, Domain ontology / upper resources (connectors). Generic API / metamodel (DCI: Form / Flow) client. Extension protocols / APIs.

Resource: Model, graph (encoding). Dataflow: order encoding. Sort dimensional measures (SortedSet?).

Encoding: Semiotic levels metaclass, class, instance, occurrence (in context in role: metaclass?).

Ontology matching: signatures encoding. Dimensional ordered aggregated measures. Sets. Semiotic reification.

Use Case: Goals App

Goals App: purpose / goals / domain driven syndication of integrated business / social / cloud application features. User / Groups / Roles Purpose(s), Goal(s), Task(s) "intelligent" tracking oriented focus providing an abstraction and integration layer of players process flows / interactions and players process assets management and semantic orchestration.

Goals App: Semantically annotated gestures / interactions (contexts, purposes messages / interactions / resources / content). Subject context occurrence role attributes values (metaclass, class, instance, occurrences).

Goals App: API Facade for rendering aggregated data roles in contexts interactions topics / subjects assets (conceptual domain contexts axis / state views / activations: Forms / Flows). Example: domain declared Customer (actor / role), Product, Order, Purchase, Invoice, etc. topics / subjects assets rendered in contexts (Sales Report, Expenses Report, etc. embedded / linked dashboards). Wizards.

Goals App: Browse / search / activate: history / relations / referrer context / interaction / gestures roles traceability / (dialogs). Gestures / interactions (actor / asset, actor / actor). Wizards.

Goals App: Hypermedia contents APIs (embedded / embeddable resources: Semantic contextual Wiki / Apache Stanbol / CMS: hypermedia augmentation, knowledge / behavior maps). Integration: augmentation / sync backends / apps. Extension: services / APIs. Annotate / augment link content. DAV protocol (integration / extension facades).

Low level Resource / Message / Context model / layers API. REST. Render DOM Context / OGM Domain (model) instances: Restful Objects / Apache Isis / HAL / GraphQL (meta / domains models endpoints) like APIs. Forms / Flows MVC / DCI APIs (connectors / clients / adapters). Wiki APIs: Knowledge discovery. Assistants.

Transactions: Request IDs. CQRS. Reified interactions / gestures. Event sourcing / logs. Reified (resource) Message activation (dataflow) Forms / Flows.

Reference Model / Encoding:

Syntax: data (signs). Semantics: schema (concepts). Pragmatics: behaviors (objects CRUD / Interactions / Contexts).

Reify able roles. Agregate semiotic Activation of a statement resources roles. Example: aggregate kinds / entities. Instance (extension) / Class (intension): Object / Concept.

Order / hierarchies: reified Context / Concept roles relation (example: Concept as Context, Concept aggregations are siblings / children of first Concept).

CSPO Reference Model: (attribute / value). Reified roles (Semiotic layer). Order. Hierarchies.

Encoding: To do. Compatible with Message Augmentation Mappings Predicate comparison and Index traversal.

Index: To do. Compatible with Message Augmentation Mappings Predicate comparison. Encoding representation (cons list, graph, tree, etc.). Truth values (flows) traversal.

Comparison: Ternary truth values results (previous / parent, current / OK, next / children) according Predicates for a Message resource set and and a Mapping comparison (apply augmentation).

Predicates: quad encoded comparison specification, result of iterating Mapping comparisons until Augmentation Message matches all Resource set. Augmentation comparison behavior. Message / Augmentation application specification.

Mappings: comparison iterations matching Message / Augmentation Predicates patterns truth values. Compare input Message resource set with Mappings till truth values are all OK. Perform Augmentation.

Model encoded Augmentation (Aggregation, Alignment, Activation) determined Comparison, Predicates, Mappings.

Encoding: IDs. Relative / contextual metadata. Serialization. Index.

Event: Message (specification), Augmentation (instance).

Predicate: comparison specification (axis). Message SPO.

(Message, LHS, Predicate, RHS);

Mapping: comparison result (patterns). Augmentation SPO. Template Transform rendered in Message context resources set.

Abstract Adapter / Connector / Client APIs:

Populate Model Context layer (Model Statements).

Protocol: Form / Flow Context (interaction). I/O data / schema / behavior input / output abstraction. Domain (model driven) interfaces / schema.

Inputs (data): Resources augmented / aggregated from OntResource until Dimensional layer.

Protocol (schema / behavior): Browse from Dimensional up to OntResource layers selecting, creating and matching intermediate Context(s). Form / Flow interaction use case roles.

Aggregation / augmentation: feedback from interactions (levels) CUD interactions.

Adapter Meta Resources (upper). Translate inputs. Render outputs from Model protocol into backends adapter specific protocol APIs.

Languaje levels: Use, mention. Pronouns.

Sign, Interpreter, Concept, Object: relations (syntax, semantics, pragmatics).

Adapter Model: metaclass, class, instance, occurrence. Context (interpreter), Sign, Concept, Object.

Connector I/O (gestures):

Client I/O (dialog):

Model layers:

OntResource: Resolves reified aligned / matched aggregated Resources.

Predicate: 'kind', aggregates roles attributes / values. Grammar.

Encoding: metaclass, class, instance, occurrence (contextual / nested / orders / ops) CSPO IDs. CURIEs.

Encoding: Sets CSPO Context specification (sets quad encoding).

Model:

(OntResource, OntResource, OntResource, OntResource);

OntResource represents aggregated / matched different identifiers / URIs referring to the same subject.

(Predicate, OntResource, OntResource, OntResource);

For a Predicate occurrence, attributes / values.

(Message, Predicate, OntResource, OntResource);

For a Message Predicate occurrence, possible attributes.

(Context, Message, Predicate, OntResource);

Occurrence (object) for a Context (interpreter) Message (sign) Predicate (concept). Adapter: Context layer (semiotic interpreter).

(Transform, Context, Message, Predicate);

(Mapping, Transform, Context, Message);

(Template, Mapping, Transform, Context);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping);

(Role, Resource, Augmentation, Template);

(Statement, Role, Resource, Augmentation); Augmentation of which Statement is result of.

(Model, Statement, Role, Resource);

(Entity, Model, Statement, Role); Model (Backends) aligned entities.

(Kind, Entity, Model, Statement);

(Class, Kind, Entity, Model);

(Flow, Class, Kind, Entity);

(Behavior, Flow, Class, Kind); Statement, proposition.

(Value, Behavior, Flow, Class); Value on which Behavior occurrence holds.

(Unit, Value, Behavior, Flow);

(Dimension, Unit, Value, Behavior);

(Measure, Dimension, Unit, Value); Truth values. Equivalent Measure(s), comparisons (order / hierarchies). Measure Dimension attributes / values.

Ontology Matching: Equivalent propositions that for different subjects have the same dimensional layer equivalent Measure(s) Value(s): attributes / values.

Example: Application (protocol) shows aggregated Measures Dimensions, select Unit / Value and assert / browse Measures.

Pick (matched / new) Behavior corresponding to Measures Values. Select available / new Flow.

Pick Flow Class and assign Kind (DCI Role). Assign / create Entity (model alignment / assignation).

Follow up in occurrences hierarchy: CRUD / CUD available / possible. Perform Augmentations.

Rendering: S: current document URL, P: link tag body, O: href, rel: Context (referrer). Navigation: GET / headers. GET (navigate, possible resources, posible contexts / subjects / attributes): CRUD / Contexts aggregation / transforms / matching. Encoding: CRUD / browse layers (CSPO Patterns Forms / Flows layers de-aggregations / faceted traversals).

Encoding:

Encoding, APIs: REST HATEOAS, JSON-LD, HAL. Distributed (normalized) address ID spaces.

TBD:

Layers down / up traversal:

C: Anchor rel (referrer);

S: Current URL;

P: Anchor tag body;

O: Anchor href;

Layers up / down traversal:

C: Current URL;

S: Anchor rel / referrer;

P: Anchor tag body;

O: Anchor href;

Functors functional declaration:

(((a: O, b: S), c: P), d: C);

(((O, S), P), C): Referring Context. Augmentations functors signature. Traversal performs functor augmentations "backwards" traversal direction concatenating type, role, interactions transforms incrementally.

Encoding (example): recursive CSPO IDs: (metaclass, class, instance, occurrence) IDs. URLs: domain/CID:SID:PID:OID. Graph URLs / rels traversal.

Addressing: URLs encode complete (possible) state flows: reified model state URLs (faceted browse / CRUD).

REST HATEOAS: Link rel (account): deposit, whitdraw, etc. Flow Behavior "referrer" rel.

Resource entity 'whitdrawal': context interaction. Actions 'possible'. Behavior Flow "referrer" rel.

ROC Resource: Resource set resolution (operations over subjects).

Reactive: PUT registers callbacks IRIs (subscribe to updates). Streams: Functors (hypermedia events / subscriptions dataflows).

Serialization / Links: JSON, XML (XPath, XLink, XPointer, XQuery, XForm). Encoding (rel, location, href, text). Framing / GraphQL.

State graph locators: relative lists identifiers encoding of all HATEOAS browsing attributes of a given state: logs.

As we have shown, JSON-LD itself is not a complete technology stack, it needs ontologies to express domain semantics. In future work we would like to investigate how a lightweight ontology to support a wide range of application domains could be modeled. Furthermore, we would like to explore various ideas to create smarter service clients.

Property graphs allow properties (key/value pairs) to be associated with both nodes and links in directed graphs. This allows you to annotate links with information such as the start and stop times for when the link is valid, its provenance, a statement about its quality and so forth.

Resource Monad: encode protocol functors. Endpoint address activation behavior facades. Graph state / rels traversal: Monad encodes entire state location flows to current CSPO URL IDs state (traceability in interaction context rels). Abstract Form / Flow attrs / rels.

Functors resolution on API addresses URLs: resource monads rels / attrs activation.

Model / Domain levels of common model / domain monads, functors: model / domain abstract augmentations / behaviors. Declared in model contexts messages / augmentation instances.

Behavior layer renders domains possible aggregated augmentations / messages of model functors composition. Rendered in domain levels as concrete contexts operations: named context operations over abstract model functors behaviors.

Adapter (Connector / Client):

Synchronization: Functional. Monads (source / dest: domain / range). Functors (APIs: templates / event drivers for function composition / translation). Inverse functions: backend IO protocols / formats. Adapter endpoint resolution: activates on backends protocols / formats / data.

Adapter (Connector / Client): Model Encoding. Container (reactive message / event driven) APIs. Model / Container APIs interactions.

Adapter / Model "statements" IO abstraction (Forms / Flows: Message events attribute / values).

Message (events): bidirectional CRUD streams (Adapter "template" methods, Model Message declarations):

Adapter: Context layer (semiotic interpreter).

onCreate;

onRetrieve;

onUpdate;

onDelete;

Ontology Matching:

Statements (encoding): Proposiciones / Razonamientos / Cuantificadores / Predicados / Clases / Relacion: validez / valor de verdad (en contextos, ejes, variables: funciones / enunciados / casos de sustitución / equivalencias). Pronombres.

Truth values: Predicate / reasoning. Sets encodings Predicate comparisons matches context templates / transforms mappings: inferences.

Relaciones de equivalencia, clases de equivalencia: reflexividad, transitividad, simetría. Identidad. Propiedades determinan alcanze / relato, dominio / codominio, universo y campo. Clases y relaciones (atributos / valores) determinan matcheos / relaciones entre clases / individuos y relaciones y tuplas / miembros. Formas proposicionales, categorícas (predicados), clases y relaciones (reglas, valores de verdad) para afirmar equivalencias (operar entre relaciones de propiedades).

Relaciones, cardinalidad: (1, n), (n, 1), (1, 1), (n, n). Función / inversa: inyectiva / biyectiva.

Relaciones de orden. Inclusión / jerarquías. Rendering: lattice (encoded statements / properties bitstring / vector). Equivalencias.

Predicate como occurrence de un Predicate: axis (attributes / values). Intensión / extensión (representante partición)

Functors:

Functors: model layers aggregations declarations / instances. Type, Role, Alignment levels. Domain / range: CSPO contexts (Template, Transform signatures). Transform: Mapping Message. Hypermedia events dataflows triggered functors (signature bindings).

Message: Functor Declaration. (events / grammar: protocols).

Augmentation: Functor Instance.

Functors functional declaration:

(((a: O, b: S), c: P), d: C);

(dado rango y alcance, universo: U de una relación: P, inferir dominio y codominio, campo: C). TBD.

From Object (O) extension / instances to Context (C) intension / class. Matching grammar shapes. Incrementally render type, role, occurrence context layers.

Type functor: contexts stream.

(((Mapping, Augmentation), Template), Resource);

Context layer class / instances.

Role functor: type contexts occurrences stream.

(((Template, Resource), Augmentation), Role);

Type Context layer class occurrence (Subject) in aggregated context layers.

Alignment functor: type occurrence attributes / values in contexts interactions stream.

(((Augmentation, Role), Resource), Statement);

Type Subject occurrence attributes / values (statements / augmentation "kinds").

Behavior flows functor composition: Behavior, Flow, Class, Kind, Entity layers aggregation. Determine type, role, alignment augmentations. Example: type (Class Model) in context (Flow Entity) in interaction (Behavior Kind).

Navigation (TMRM, key / value) functors:

Keys.

Remote keys.

Local values.

Key / value proxies.

Value proxies.

Constraints.

Merge.

Legends (constraints).

Path languages.

Path expressions.

Encoding:

Encoding: metaclass, class, instance, occurrence (contextual / nested / orders / ops) CSPO IDs. CURIEs.

Encoding: Sets CSPO Contexts specification (sets quad encoding).

Encoding: Functor application. Predicate: functor behavior, domain: statement predicate, transform / range: statement object.

Encoding: Levels (OntResource context hierarchy) reification: Message as Predicate, etc. Resource Monad (context statement / signatures). Functor aggregation: levels (type, role, alignment).

Encoding: Grammars. OntResource hierarchy reification: rules (contexts) / non terminals (reified Predicates / Kinds). Aligned OntResource URLs: terminals. Augmentations: productions (functors).

Link Grammars. Types: links left / right types defined when a shape / slot match satisfaction occurrs (roles).

Parsing: extract propositions, knowledge assertions (in a domain ontology). Assert propositions links, order, concepts relations (between domains). Link Grammar. ISO TMDM / TMRM.

Parsing: extract prescriptions, knowledge rules (in domain ontology concepts relations: causal, requirements, etc.). StratML.

Parsing: infer possible statements propositions / prescriptions productions. Link Grammar Disjuncts. Embeddings.

Reference model (encoding):

Statements:

(OntResource, OntResource, OntResource, OntResource);

Grammar dictionary: link type, left, context (word), right types:

(Predicate, OntResource, OntResource, OntResource);

Grammar Links (Functor / Parser): link types, left / right links.

(Message, Predicate, OntResource, OntResource);

Types / Categories: Resource Monad. Examples: aggregate Predicate / Entity Kind / Resource Type Attributes / Values. Infer SPO statements.

Functor / Parser signature:

(((O, S), P), C);

Model layers:

OntResource: Resolves reified aligned / matched aggregated Resources.

Predicate: 'kind', aggregates roles attributes / values. Grammar.

(OntResource, OntResource, OntResource, OntResource);

(Predicate, OntResource, OntResource, OntResource); For a Predicate occurrence, attributes / values.

(Message, Predicate, OntResource, OntResource); For a Message Predicate occurrence, possible attributes / values. Functor declaration.

(Context, Message, Predicate, OntResource); Occurrence (object) for a Context (interpreter) Message (sign) Predicate (concept). Adapter.

(Transform, Context, Message, Predicate);

(Mapping, Transform, Context, Message);

(Template, Mapping, Transform, Context);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping); Type Functor Augmentation instance.

(Role, Resource, Augmentation, Template); Role Functor Augmentation instance.

(Statement, Role, Resource, Augmentation); Augmentation of which Statement is result of. Alignment Functor Augmentation instance.

(Model, Statement, Role, Resource);

(Entity, Model, Statement, Role); Model (Backends) aligned entities.

(Kind, Entity, Model, Statement);

(Class, Kind, Entity, Model);

(Flow, Class, Kind, Entity);

(Behavior, Flow, Class, Kind); Statement, proposition.

(Value, Behavior, Flow, Class); Value on which Behavior occurrence holds.

(Unit, Value, Behavior, Flow);

(Dimension, Unit, Value, Behavior);

(Measure, Dimension, Unit, Value); Truth values. Equivalent Measure(s), comparisons (order / hierarchies). Measure Dimension attributes / values.

Update:

Distributed Integration and Consistency for Knowledge Semantic Interoperability.

Link grammar: (employer, employee). Categories, monads. Reference Model.

Link Grammars. Types: links left / right types defined when a shape / slot match satisfaction occurrs (roles).

Parsing: extract propositions, knowledge assertions (in a domain ontology). Assert propositions links, order, concepts relations (between domains). Link Grammar. ISO TMDM / TMRM.

Parsing: extract prescriptions, knowledge rules (in domain ontology concepts relations: causal, requirements, etc.). StratML.

Parsing: infer possible statements propositions / prescriptions productions. Link Grammar Disjuncts. Embeddings.

Functors: model layers aggregations declarations / instances. Type, Role, Alignment levels. Domain / range: CSPO contexts (Template, Transform signatures). Transform: Mapping Message. Hypermedia events dataflows triggered functors (signature bindings).

Message: Functor Declaration. (events / grammar: protocols).

Augmentation: Functor Instance.

Functors functional declaration: (((a: O, b: S), c: P) d: C);

(dado rango y alcance, universo: U de una relación: P, inferir dominio y codominio, campo: C).

From Object (O) extension /instances to Context (C) intension / class. Incrementally render type, role, occurrence context layers.

Type functor: contexts stream. (((Mapping, Augmentation), Template), Resource). Context layer class / instances.

Role functor: type contexts occurrences stream. (((Template, Resource), Augmentation), Role). Type Context layer class occurrence (Subject) in aggregated context layers.

Alignment functor: type occurrence attributes / values in contexts interactions stream. (((Augmentation, Role), Resource), Statement). Type Subject occurrence attributes / values (statements / augmentation "kinds").

Types / Categories: Resource Monad.

Navigation (TMRM, key / value) functors:

Keys.

Remote keys.

Local values.

Key / value proxies.

Value proxies.

Constraints.

Merge.

Legends (constraints / Mappings Augmentations).

Path languages.

Path expressions.

Encoding. Model layers:

Levels: reify layers from bottom up through contexts hierarchy superclass contexts relationship. Message, Type, OntResource, etc.

Resource context: Resource monad CSPO Form. From Message / Type OntResource / Resource value mappings (functors / navigation).

OntResource context: (Resource, Occurrence, Attribute, Value) Form. Aggregated from Resource context.

(Resource, Resource, Resource, Resource);

(OntResource, Resource, Resource, Resource);

Values. Sets. Equivalences / matching assertions. In Context occurrences, matching attributes / values (recursion to contexts / occurrences).

(Type, OntResource, Resource, Resource);

Labels. Type (C: key) occurrence (S: value), value occurrence (P) aggregated attributes (O).

(Message, Type, OntResource, Resource);

For a Message (proxy / subject context), Type (sign / key / label), OntResource (concept: P), Resource (object / value). Functor declarations. TMRM navigation / paths.

(Context, Message, Type, OntResource);

Context (key / value interpreter / map: connectors legends), Message (sign / proxy / subject), Type (concept / key / label), OntResource (value). Key / Value Adapter. TMRM.

(Transform, Context, Message, Type);

(Mapping, Transform, Context, Message);

(Template, Mapping, Transform, Context);

(Augmentation, Template, Mapping, Transform);

(Resource, Augmentation, Template, Mapping);

Type Functor Augmentation instance.

(Role, Resource, Augmentation, Template);

Role Functor Augmentation instance. Role: CSPO Resource role type in occurrence / context.

(Statement, Role, Resource, Augmentation);

Augmentation of which Statement is result of. Alignment Functor Augmentation instance.

(Model, Statement, Role, Resource);

(Entity, Model, Statement, Role);

Model (Backends) aligned entities.

(Kind, Entity, Model, Statement);

(Class, Kind, Entity, Model);

(Flow, Class, Kind, Entity);

(Behavior, Flow, Class, Kind);

Statements: propositions, prescriptions / rules / productions. DCI / Link Grammar. Entities Statement occurrences: contexts / interactions / proposition Class / Kind roles (satisfaction / rules).

(Value, Behavior, Flow, Class);

Value on which Behavior occurrence holds.

(Unit, Value, Behavior, Flow);

(Dimension, Unit, Value, Behavior);

(Measure, Dimension, Unit, Value);

Truth values. Equivalent Measure(s), comparisons (order / hierarchies). Measure Dimension attributes / values.

Ontology matching: sets, singletons / equivalence classes merge. Encoding: Sets CSPO Contexts specification (sets quad encoding).

Proof of concept: Relational inductive biases, deep learning, and graph networks. Deep Graph Infomax. Train model to extract (augmented) knowledge from training set encoded models. Test output in new encoded examples / different ontologies / domains.

Encoding of types, instances and behavior flows along with the data and information needed to infer knowledge necessary to understand encoded models (meta models).

Model (update / draft). TBD.

Backend: Kernel. Models, Services / Adapters "service" URLs (naming schemes). Messages. Interfaces signatures / content types. JMX, OSGi, CDI, Spring, MQ / Camel (events / bus). Reactive facade for protocol endpoint / service interactions. Dialog (Form / Flow) interaction contexts.

Pluggable "bundles" adapter implementing APIs (i.e.: index, naming, registry services) for each (declarative / functional / reactive) Model / Service / Adapter bundles. HAL (Augmentation / Legends), ActivityStreams, StratML, LoD, Solid Adapters.

Context hierarchies: categories of wrapped subject occurrences. Statement Context monad (category) for Subject occurrences.

Context category instance identity: Employee Kind, Work Behavior, etc. Context statement attribute / value aggregation. Key / value abstraction scoped in category type / instance.

Functors: Layers (Context categories) aggregation / transforms. Augmentations.

Augmentations: Categories Aggregation (types), Alignment (contexts), Activation (roles / matching in interactions).

Locators (keys): (metaclass, class, instance, occurrence);

Metaclass: Transform OntResource

Class: Message Augmentation

Instance: Statement Role

Occurrence: Flow Kind

Locators (keys): local / remote keys. Navigation. Paths. Legends. Signatures (streams).

Monadic  (locators / signature streams based) category wrappers Functors:

M<T>::flatMap(functor(T) : U) : M<U>;

Dado rango y alcance, universo: U de una relación R, inferir dominio y codominio, campo: C.

From Object (O) extension /instances to Context (C) intension / class.

Aggregation (types) functor signatures: stream.

Alignment (contexts) functor signatures: stream.

Activation (roles / matching in interactions) functor signatures: stream.

Encoding. Model layers:

URIResource context: CSPO form. RESTful / HAL monad: HTTP category functors.

OntResource context: (Resource, Occurrence, Attribute, Value) form. Aligned (matched) URIResource(s).

Resource context: OntResource (aligned / matched URIResource) occurrences in reified Role in Statement.

(URIResource, URIResource, URIResource, URIResource);

(OntResource, URIResource, URIResource, URIResource);

Aggregated URIResource OntResource attributes / values (recursion to attributes / values OntResource).

(Transform, OntResource, URIResource, URIResource);

(Mapping, Transform, OntResource, URIResource);

(Template, Mapping, Transform, OntResource);

(Augmentation, Template, Mapping, Transform);

(Message, Augmentation, Template, Mapping);

(Context, Message, Augmentation, Template);

Model.

(Resource, Context, Message, Augmentation);

(Role, Resource, Context, Message);

Reified CSPO / Resource, Occurrence, Attribute, Value Resource role types in Resource occurrence / context.

(Statement, Role, Resource, Context);

(Entity, Statement, Role, Resource);

Aggregated "subject" occurrences of Resource in Role in Statement(s).

(Class, Entity, Statement, Role);

Aggregated Entity Role occurrences type (attributes).

(Kind, Class, Entity, Statement);

Aggregated kinds / roles ("interfaces") of Class occurrences.

(Flow, Kind, Class, Entity);

Action "instance". Entity of Class performs role (Kind) of Behavior Flow.

(Behavior, Flow, Class, Kind);

Action "class". Statements: propositions, prescriptions, rules, productions. DCI / Link Grammar. Context satisfaction (rules).

(Measure, Behavior, Flow, Class);

(Unit, Measure, Behavior, Flow);

(Dimension, Unit, Measure, Behavior);

Order.

Encoding:

Radix, Digit, Position. SPO: radix:digit. Radix: resource count. Digit: resource radix index. Position: context hierarchy aggregation. Normalized radix encoding: metaclass, class, instance, occurrence context metadata operable IDs. Example: Base 2 normalization, concepts lattice bitstring (routing tables), boolean operations metadata extraction / traversal (contexts masks). IPv6 addresses. Masks (contexts), routes. Address resolution resolves recursively expanded (CSPO hierarchy contexts / occurrences / attributes / values) resources representations (graph statements members nested browseable addresses of ID contexts). Addresses representing behaviors (HATEOAS: browse representation links / addresses renders state / operations, i.e.: address plus according referrer context and representation addresses links states CRUD operations).

Normalization: contextual form IDs. Internal four sides polygon angles for each CSPO addressable IDs. Embeed operable metadata in CSPO contextual IDs. Vector space model like.

CAM (Ternary bitstring). Routes. CSPO Functional Mappings (contextual / occurrences vector IDs).

Addresses browse state transforms: navigation renders context layers statements transforms with contextual browsing state (IDs, referrer, contextual / occurrences vector IDs, metadata) as parameters. Contexts navigation (Forms, Flows) as functors. Navigation state transforms: possible activations context / argument / attributes / values. Dialog / prompts (argument resources navigation state transforms).

JAF / Naming / Registry (HATEOAS Forms / Flows navigation / states): DCI / MVC Engine.