DOM OntResource: Models Meta Meta Model. Reference Model. Runtime / dataflow APIs.

Parsing: AST Object URL monad layers hierarchy. Parser combinators. Monad Zippers (tree encoding). Order. Flows (streams). Parser combinators.

OntResource:

Reactive (producer / consumer) entity. Hierarchy templates determines signatures / dataflows (topics / queues).

Signatures: observer (topics): ObjectID events stream. URL events stream. Role events stream. Layers hierarchy events set (reified: Resource, Statement, Kind, Relation events). Filter. Template / abstract methods. Core OntResource API behaviors. Dataflow.

Signatures: observer (topics): ObjectID events stream. URL events stream. Role events stream. Layers hierarchy events set (reified: Resource, Statement, Kind, Relation events). Filter. Template / abstract methods. Core OntResource API behaviors. Dataflow.

OntResource abstract APIs:

Abstract: Meta Model objects occurrences (aggregated by Object URL and Roles):

OntResource::ObjectID;

OntResource::Occurrences<URL, Role> : OccurrenceID;

OntResource represents aligned Objects / Resources of which the same Object (ObjectID) occurs in different Roles and (possibly) diferent URLs. Matching and alignments by reifying layers into abstract resources / OntResources.

Meta Model(s) APIs (Augmentation, Dataflow, Dimensional) reflects reference model events reacting to and producing signature matching messages on Meta Model(s).

OntResource's Roles are the DOM (Dynamic Object Model) types of the Meta Model(s) layers Occurrences / Objects / Contexts / SPOs. FCA Lattice contexts occurrences metadata. DOM APIs (reifying hierarchy layers, abstract templates):

Role: occurrence / object in CSPO slots. Denotes resource types in positions in statements (i.e.: Kind in Relation). Role CSPO is object / occurrence in statement occurrence position, Role type (i.e.: Kind, Relation) stated as Role instances in Meta Models with corresponding Kinds for its complimentary CSPO resources.

(Instance, Class, Metaclass, Occurrence);

(Context, Occurrence, Attribute, Value);

Meta Model(s). Augmentation APIs hierarchy:

Object<URL> : OntResource;

Abstract. OntResource aligned URL / template method design pattern.

Object<URL>;

Abstract. ObjectID.

Occurrence : Object, Object<URL> : OntResource;

Abstract ObjectID, OccurrenceID. Reify aggregated hierarchies.

Role : Occurrence, Occurrence, Object<URL>;

(Resource : Role, Role, Occurrence, Object<URL>);

(Statement : Resource, Resource, Role, Occurrence);

(Kind : Statement, Statement, Resource, Role);

(Relation / Model : Kind, Kind, Statement, Resource);

Class hierarchy APIs:

(Instance, Class, Metaclass, Occurrence);

(Context, Occurrence, Attribute, Value);

Meta Model(s). Dataflow APIs hierarchy:

(Augmentation, Mapping / Predicate, Transform / Template, Resource);

Meta Model(s). Dimensional APIs hierarchy:

(Dimension, Unit, Measure, Value);

Meta Model(s). Semiotic APIs hierarchy:

(Context, Sign, Object, Value);

Domain Models. Service APIs. Templates.

Dataflow, Dimensional and Domain Models declaratively stated in root Meta Model.

Messages. Encoding / I/O:

(Class, Instance, Member, Value);

Augmentation:

Domain (Templates)

Meta Model (Activation)

Meta Meta Model (Augmentations

Meta Model (Activation)

Domain (Templates)

Encodings:

(a (b (c (d, nil) : First / Rest binary tree.

(K: (K: inst, V: cls), V: (K: mcls, V: occur))

Object / Occurrence: Reify layers Object / Occurrence keys / values. Reference Model along with OntResource.

Message I/O:

(Class, Inst, Attr, Val);

Templates: I/O Dataflow (aggregated messages) declarative bindings. Aggregated message facade (Augmentation, render / apply layer entity schema / data). Forms / Flows.

Forms / Flows: HATEOAS HAL. MVC. REST. Meta Models DCI based protocol.

HATEOAS: Forms / Flows Operations / Dataflow Representation / State IO (CRUD) prototypes / templates. Dialog. Prompts. Gestures. Context: navigation state (i.e.: pick operation value prompt shows value type Forms). DDD DOM.

Models Sets Reification (Populate DOMs / FCA):

OntResource

Object

Occurrence

Role : CSPO hiers Sets.

Resource : Role Set member.

Statement : Set members Role aggregation.

Kind: Statement Resources aggregations. Roles intersection sets.

Relation: Kind Statements aggregations. Transform: Kind Resources related to themselves (ID), then Relations to other Resource via Dataflow Kinds domain / range relationship (ordered).

Relation: aggregated aligned entities. Views (transforms). Kind members occurring in Statement Resource(s). Functors / Monads:

Relationship<A : Relation>::flatMap(F : Function<A : Relation, B : Relation>) : Relationship<B : Relation>;

Entity<A : Relationship>::flatMap(F : Function<A : Relationship, B : Relationship>) : Entity<B : Relationship>;

Function: declarative dataflow transform.

Dataflow Kinds domain / range: Grammar. Reify Kinds as CSPO and assert Statement. Aggregate further Kinds (until primitives).

Valid Statement (Grammar / Relation): domain / range, CSPOs backing assertions apply.

I/O: Parse / aggregate input Statements into corresponding Roles / Resources. Aggregate / match Kinds. Relations: render / activate. Resolve output Statements.

Built in Relation(s): ID, equals, inverseOf, parent, child, previous, next, etc. (upper ontology / meta model). Composites: Monad Zippers.

Meta Meta Mode (TBD)l:

DOM: OntResource, Object, Occurrence, Role. Express Meta Model levels metaclass, superclass, class, instance, occurrence relationships.

(Superclass, Class, Occurrence, Instance);

Metaclass: Role.

Occurrence: Role instance.

**Ontology Matching:**

Data, Schema, Behavior matching / alignment.

Data: keys / values.

Schema / Information: relation tuples rows.

Behavior / Knowledge: relation tuples rows data / information flows (dimensional).

FCA Augmented Resources.

Relationship<Relation[]> Monad. Relation[]: selector, Relations which are instances wrapped by this Relationship scope.

ToDo: Relation selector API.

Relationship Monad (instantiated from Relations) functions: assert(e1 : Entity) : Function<Entity, Entity>;

ID Relationship Monad:

Relationship Monad instance wrapping Relations of selected (predicates) identical objects (Resources).

Relationship Monad assert dispatch:

Invoke Entity assertion matching logic against each individual Relationship Relation and collects results.

anEntity.flatMap(anId.assert(e1 : Entity) : e2 : Entity (anEntity if same Entity, previous / next Entity if not same Entity).

ID previous / next Entities in Relationship Relation complimentary (Entity comparison complements) axes.

Entity Monad built in Relationships: ID, equals, inverseOf, parent, child, previous, next. Apply Relationship assert in the same manner than ID. Logical browsing. Streams (Relationship ordered Entity iterators).

Domain Models Entities / Relationships: transforms of underlying Entities given Relationships contents.

Relationships (upper domain): before, during, after, cause, effect, implies, etc.

Relationsip assertions reified / parsed as / from Relation Statements (Messages statements predicates).

Entity Relationships rendered / parsed as / from Relations, Kinds, Statement, Resources (Message contents).

**Backend Architecture (To Do):**

XML: XSL, XPath, XLink, XPointer, XQuery, XForms: hypermedia addressing / state / flows encoding / Message endpoints protocol.

Models.

Messages.

Encoding.

Endpoints.

Protocol.

Domain Connectors.

APIs.

Models:

Dispatch to Model layers context resources streams. Message IO. Endpoint Message matches in Model context: activation (Dataflow).

Messages:

Meta Meta Model entities (Relation). Meta Model layers scoped context statements.

Encoding:

Model layers scoped context statements.

Endpoints:

Model layers context resources streams (pub / sub topics). Topics: OntResources(s). Signatures: Dataflow Message IO wiring.

Protocol:

Message Dialog: i.e.: XML encoded Context statements Message IO with Model layer scoped prompts, placeholders, wildcards, variables. Models. Messages. Streams.

Domain Connectors:

Tools. Service Resource: URL, streams (Messages I/O).

Tools:

NakedObjects / Apache Isis. Apache MetaModel. JBoss Teiid. JDBC. JCA. Apache Stanbol. Apache Clerezza. OData / OpenAPI. JSON-LD. Spring HATEOAS / HAL. Apache Any23. D2RQ. R2RML.

Message Matching:

FCA Augmented Models: Context objects / attributes: layer quad resources Role, Kind, Resource.

Resource Context Concept: Resource x Kind.

Resource Context Object: Role x Resource.

Model / Schema Matching:

FCA Resource Context Concept.

Data / Resource Matching:

FCA Resource Context Object.

Matching Productions:

Result Set (query / augmentation result).

Augmentation (aggregation of new statements, alignment of new knowledge, activation of transforms / flows: result set).

Relationship / Entity monads mappings results.

FCA Scaling: Role > Kind > Resource aggregation of matching objects / attributes.

FCA Augment Resources: Role, Occurrence, Object (Resource), Concept, Object, Kind. Grammars. Match schema, instances. Mapping transforms: match behaviors.

FCA Contexts from Sets aggregation:

Set Roles: Context, Subject, Predicate, Object.

Sets aggregation: Statement, Kind (SuperKind, Kind, Attribute, Value) Attributes: class / Values: metaclass, Resource (Meta Model Roles: Kind context, Resource SPO), Context (Relation). Reified Kinds.

Aggregation streams: Sets reactive events aggregation. Sets (ordered) description APIs.

Sets aggregation: FCA Contexts scaled objects / attributes from Sets aggregation. FCAAPI.

**Deployment:**

Apache MetaModel. JBoss Teiid. Connectors (I/O). APIs: Model Services (reify data, schema, behavior alignment in Connectors data structures). OpenRefine Knowledge (data, schema, behavior) alignment extensions (Model Services APIs).

**APIs:**

Contexts (DCI / HAL / HATEOAS):

Context Guided Data augmented (contextual hypermedia) Interactions. Wizards APIs. XForms: rendering (REST HATEOAS).