Resources metamodel

A triples (quads) input graph is processed and aggregated into metamodel levels layers.

All metamodel levels: Facts, Objects, Purposes (layers) statements and helper resources are encoded into quads following the same pattern, from source SPOs to behavior (Purpose) quads.

An upper layer takes its base layer statements as subjects of their own statements. Then it takes base layer Kinds (types) as its predicates. Finally, base layer SPOs are layer's objects.

Each layer statements, types and individuals (models) are represented by the following metamodel classes. The rest of the schema is described by instances of the following classes.

Class Predicate.

- holds(resource : Resource)

Class Set.

- predicate : Predicate

Class Resource (extends Set).

- URI : String
- resourceId : Resource
- context : Resource
- subject : Resource
- predicate : Resource
- object : Resource
- parents : Map<Resource, Resource>
- roles : Map<Resource, Resource>
- mappings : Map<Resource, List<Resource>>
- previous(ctx : Resource, state : Resource) : List<Resource>
- next(ctx : Resource, state : Resource) : List<Resource>
- children(ctx : Resource) : List<Resource>
- add(res : Resource, role : Resource)
- list(role : Resource) : List<Resource>
- apply(template : Resource) : Resource
- owlRdfResource : Node

Resource dataflow, callbacks, events, messages, Templates. Add resource (res : Resource, role : predicate/set). Children collection derived from parent. Update.

Statements, Kinds, CSPOs, Resources

Each metamodel level has its own meanings for their statement components. The most important helper resource derived from statements are Kinds. Kinds are a form of type inference based on a resource attributes and values. A given SubjectKind, for example, is a 'kind of' Subject.

A Subject has, given this model, some parent statements for which its role is Subject (resource model instance) and plays a SubjectKind.

A SubjectKind aggregates Subjects sharing the same set of Predicates in different statements inferring their 'class'. It then can aggregate this Kind with statements with the same attributes (Predicate) with those with the same values (Object) inferring their 'metaclass'.

The same accounts for PredicateKinds and ObjectKinds with their respective attributes and values. TBD.

Resources metamodel levels

The different metamodel layers, aggregated as stated before, are the following:

The Facts layer attempts to arrange input statements and their components into sets: Facts (actual statements), Subject, Predicate and Object sets and their respective Kinds. It also manages statement contexts. TBD.

The Objects (semiotic) layers arranges previous (Facts) layer SPOs into 'Signs', Kinds into 'Concepts' and statements into 'Objects'. Its statements (Topics) are of the form:

(ctx : Topic) (object) (concept) (sign)

Its Kinds are: TBD.

The Behavior (purpose) layer arranges previous (Objects) layer OCSs into 'Players', OCS Kinds into 'Roles' and statements (Topics). Its statements (Purposes) are of the form:

(ctx : Purpose) (topic) (role) (player)

Its Kinds are: TBD.

Facades are resources that group related resources (Templates). They may be considered as another model layer. For example all Facts for a given subject, all order Objects, all order management purposes. Their statements are of the form:

(ctx) (Purpose) (Object) (Fact)

Resources model instances

For each model level a similar structure of sets of statement, kinds, and SPOs is represented by actual instances of resources 'modelling' each layer entity types. Each type represents a similar 'role' in each layer. For example, Facts, Objects and Purposes are all kinds of statements as objects, signs and players are all objects (role) in statements.

Contexts, Models: Facts, Objects, Purposes.
Statements: Fact, Topic, Purpose.
Subject, Object, Topic.
Predicate, Concept, Role
Object, Sign, Actor.

Same (aggregated) instances may be shared across metamodels. Each resource (set) definition will match appropriate instances.

Resources functional mappings

Each resource has resolution of the entities corresponding itself in respect to other resources or contexts:

(Triple, Kind) : SPOs
(Triple, SPO) : Kind
(Kind, SPO) : Triple
(S, O) : P
(P, O) : S
(S, P) : O
(SK, OK) : PK
(PK, OK) : SK
(SK, PK) : OK

ID Res: common superset, common rels / links.

Data: SPO / Resources (reifying types & behavior).

Types: OCS / Kinds.

Behavior: TSP / Statements (grammars).

Resources reference model

Reference model attempts to provide an uniform manner of accessing resources metamodel and applying transformations (Templates).

This is mainly achieved by sorting hierarchicaly and horizontally a graph of resources in specific contexts.

Methods of resource are: parent, children, previous and next. Example. TBD.

anEmp.next(Emp, Dev) : Leader.

aTask.parent(aProject) : aContract.

Kinds operations (reification, hierarchies)

Kinds (types) may be reifyied as their corresponding SPOs as means to establish type hierarchies or as a metadata facility for augmenting their meaning.

Also, using reifyied Kinds there could be expressions matching resources with a given Kind, common supertypes or common links and relations.

Resource IDs

Given a resource, this concept aims to populate an 'ID' augmenting it with all resource's occurrences and the Kinds in that occurrences context.

Then, given a proper encoding, a resource 'ID' may serve to match or find similar (or equivalent) resources and query information between them. Also and expression can be built to retrieve a desired pattern of resources.

Resource ID: TBD.

Resource ID (Subject): (S.resourceID, P, O) for all statements of S.

Primitive terms.
Opposites.
Negation.
Inverse.
Complements.

Term has term in role in context. Classes.

Terminal has terminal as non-terminal in production/rule.

non-terminal has non-terminal as terminal in production/statement.

Resource ID: Template. Terminals: SPOs, non-terminal: Kinds. Statements, ordered by statement context:

(context, role) (rule, non-terminal) (statement, terminal)

Context / Role (classes) calculated through aggregated term identifiers.

Resource ID statements context: temporal metadata. Reify events (resources), temporal relations (compare) other resources/events. Truth values.

Grammars

Grammars are models built from actual models Kinds, using Kinds as SPO, respectively, in statements ('rules'). For a grammar:

Kind: non terminal
SPOs: terminal
Statements (of kinds): production rules

Templates / Protocol

A Template is a set (graph resource) of statements with any layer SPOs, Kinds, Statements, variables and wildcards. A Template may give form to a Facade and be the means of interaction with resources apply(ing) them as transformations.

Also, identity resolution (align and merge) and ordering (temporal and arrangement) can be expressed in term of Templates. Resource IDs and Template resources are the basis for dataflow implementation

Once submitted to a resource (apply) a Template starts a dialog in a 'protocol' with request / response cycles in which each part asks / replies till no resolutions left.

Example: TBD.

Facades

Facades are models narrowed to an specific context in an specific level. Facades for Facts regards to one particular Subject (anOrder) Facades for Objects regards to a particular Topic (orders) and Facades for Behavior regards to one particular Purpose (orderManagement).

Facades: models with TSP, OCS, SPO statements. Dataflow. Resource.apply(Resource).

Identity resolution / merge

Use grammar and mappings to create all possible statements between graphs to merge. Calculate Resource IDs. Merge matching IDs resources. Resolve ambiguity using functional mappings and reference model.

Identity resolution: all possible statements, all possible Resource IDs. Merge with actual resources, refine possibilities with Template dialog (contexts, this, that, theirs 'pointing' variables) mappings / reference model.

Contexts: reified contexts (identity, ordering). Truth values (statements, resources) in contexts. Reifyied ordering (events, flows, rules types/instances) temporal/contextual 'holds' values (octal). Merge Template: tautology. Dictionary: sameAs.org. Term in context (actor, role, topic).

Attributes and links discovey

(dept) (leader) (peter)
(joe) (worksAt) (dept)
: (joe) (boss) (peter)
TBD.

Ordering and temporal alignment

A series of statements like this may be presented:

(someone) (wash) (car)
(someone) (takesOut) (car)
(someone) (takesIn) (car)

Infer correct ordering. Encode ordering kinds: (actual) (prev) (next). Operate over ordering: encode position in three bits. TBD.

Order inference: opposites, complements (truth values, contexts). Order 'kinds': actual, prev, next (Subject temp context SPOs). Events, Flows, Rules (verb action, passion, state). Bounds.

Dimensions, axes, units, measures.

Primitive terms.
Opposites.
Negation.
Inverse.
Complements.

SK / OK: Complements. Ordering. Statements reification into SPOs.

PK Inverse: swap S/O.

Opposites: P respect to PK, three possible values: original value, opposite, negation. Expression. TBD.

take out/wash car complement of wash/take in car. take out/take in opposites. Opposites: order SPO. Predicate: statement with P of both kinds. Determine S (first) by PK state/ordering (Resource IDs). TBD.

Deployment features

TBD.

Adapters

TBD.

Ports

TBD.