

CDL Model and Syntax

This section provides a model of the concept or concept structure and its grammar rules of the concept description. The specification is common to all CDLs.

Concept or concept structure is modeled by semantic network as follows,

Node : Node consists of Entity. Node has two kinds. One is simple node which has no inner structure. The other is hyper node which has inner network structure. The former node is called “ElementalEntity”, the latter is called “CompoundEntity”.

Arch : Arch is a labeled directed binary relation or triple which has two nodes and one relation. Arch has two kinds. One is simple arch which has no inner structure. The other is hyper arch which has inner networkstructure. A relation of the former arch is called “ElementalRelation” and a relation of the latter arch is called “CompoundRelation”.

Reduced arch and **reduced node** : EntityConcept and RelationConcept has additional information called attribute-value pair. That is a reduced form of a simple relation and a simple node.

Definition of concepts are stored in a Concept Definition Dictionary (CDD). CDD has hierarchy of structure of concept. Every top level of concept structure of CDLs has following hierarchy,

- Concept**
 - Entity**
 - ElementalEntity**
 - CompoundEntity**
 - Relation**
 - ElementalRelation**
 - CompoundRelation**
 - Atrtribute**

1. Textual representation

There are two methods in the representation of conceptual network. One is a graphical or diagrammatic representation and the other is textual representation. In the

following syntax of textual representation is shown using the meta symbols “:=”, “|”, “Term”, and “Nil”.

Concept:= Compound Concept | Elemental Concept | Conceptual Text | Literal Concept
Compound Concept:= { Instace Label Concept Label Attribute-Value Pair...; Concept... Arch... }
Elemental Concept:= { Instace Label Concept Label Attribute-Value Pair... }
ConceptualText:= <Text> | <Instace Label Text>
LiteralText:= “Text” | “Instace Label; Text”

Instance Label:= #character string
Concept Label:= Conceptual Label in CDD | dictionaryID.Word Meaning Label
Attribute Value Pair... := Attribute Value Pair Attribute Value Pair... | Nil
Attribute Value Pair:= Attribute = Attribute Value
Attribute:= Conceptual Label in CDD | Character_String_Starting_with
_ _alphabet_kana_kanji_character
Attribute Value:= Attribute Values in CDD | Arbitrary Charactestring
Concept... := Concept Concept... | Concept
Arch... := Arch Arch... | Arch
Arch := [Instance Label1 Instance Label2 Instance Label3]

The scope of the instance_label has two modes which are open scope and close scope. In this section, open scope is default mode.

2. Visual representation

The multiple conventions of representations are supplied for easy understanding of the concept structure. In this section typical examples are shown. Any other conventions will be permissible for the purpose.

(1) Arch

Conventions for single arch or multiple arches.

(1-1)

[L1 L2 L3] ⇒ [L1–L2→L3] or [L3←L2–L1]

; In case of ASCII characters, use “-” and “->” instead of “–” and “→”.

(1-2)

$$\left. \begin{array}{l} [L1 L2 L3] \\ [L1 L4 L5] \\ [L1 L6 L7] \end{array} \right\} \Rightarrow [L1-L2 \rightarrow L3, -L4 \rightarrow L5, -L6 \rightarrow L7]$$

(1-3)

$$\left. \begin{array}{l} [L1 L2 L3] \\ \text{and} \\ [L3 L2 L1] \end{array} \right\} \Rightarrow [L1-L2-L3]$$

(2) Replacement of the instance label in the arch by concept label

The instance label in the arch can be replaced by the concept label in just case of which concept is ElementalConcept, ConceptualText and LiteralText. If the instance label occurs multiple, one of the instance labels is replaced. If the instance label occurs single, it is replaced by contextual text and literal text abbreviated of the instance label. The replacement is restricted to the arch and concept in the same scope.

Special conventions

(2-1) ElementalConcept without attribute-value pair

In the case of ElementalConcept without attribute-value pair, instance label is moved to the suffix of the conceptual label.

$$\{\underline{\text{Instance label}} \ \underline{\text{Conceptual Label}}\} \Rightarrow \underline{\text{Conceptual Label Instance Label}}$$

Example:

$$\{\#A \text{ computer};\} \Rightarrow \text{computer}\#A$$

(2-2) Replacement of instance label by conceptual label

In the case of ElementalConcept with unique concept, the unique concept replaces the instance label of the ElementalConcept. The global standard IDs such as numeric concept, URL, mail address and ISBN etc. and ID specified in the CDLs (Relational Concept Label with three alphabet characters) are the typical examples.

$$\{\underline{\text{conceptual label conceptual label}}\} \Rightarrow \underline{\text{conceptual label}}$$

Examples:

$$\{\#256 \ 256;\} \Rightarrow 256$$

$$\{\#ISBN9-876-54321-0 \ ISBN9-876-54321-0;\} \Rightarrow \text{ISBN9-876-54321-0}$$

(2-3) Compound concept in case of the instance label being conceptual label

When an instance label is temporarily used in a scope, the instance label constructs CompoundConcept by replacing conceptual label by itself. Further the instance label is not referred from other arches, the instance label and semicolon are neglected. When

the instance label is referred only once in the arches, the instance label in the arch is replaced by the representation neglecting of the instance label and semicolon.

{ Instance Label Instance Label; Concept... Arch...}

⇒ { Instance Label; Concept... Arch...}

Further,

⇒ { Concept... Arch...}

(3) Concept represented by Compound concept

Compound concept to be represented by hyper node is modeled by two ways.

(3-1) Concept model to be a whole compound concept

(3-2) Concept model to be modified by compound concept

Examples:

1) < A report was received that yesterday computer was purchased.>

↓

{#A event;

<#B;yesterday computer was purchased>

<#2;report>

<#3;received>

[#2-cnt→#B, #3-obj→#2]}

↓

{#A event;

{#B event;

<#B1;yesterday>

<#B2;computer>

<#B3;purchased>

[#B3-obj→#B2, #B3-tim→#B1]}

<#2;report>

<#3;received>

[#2-cnt→#B, #3-obj→#2]}

#B represents a whole compound concept. The example occurs in the “that clause” in natural language sentence.

2) < A computer which was purchased yesterday was set up.>

↓

{#A event;

<#B;computer which was purchased yesterday>
 <#2;was set up>
 [#2-obj→#B]}

↓

{#A event;
 {#B event;
 <#B1;yesterday>
 <#B2;computer>
 <#B3;purchased>
 [#B3-obj→#B2, #B3-tim→#B1]}
 <#2;set up>
 [#2-obj→#B2]}

Concept modified by compound concept occurs in a relative clause.

3. Schema

(1) EntitySchema

(1-1) Elemental_Entity Schema

{#Concept Label E-EntitySchema Attribute Value Pair...;
StructureDescription
ExplanationDescription
ConstraintDescription
Arch...}

StructureDescription := {Instance Label SET Attribute Value Pair;
AttributeSchemaReference...}

AttributeSchemaReference... := AttributeSchemaReference
AttributeSchemaReference...
 | Nil

AttributeSchemaReference :=
 {Instance Label Attribute ref=#A-SchemaConcept Label
Attribute Value Pair}

ExplanationDescription := Concept Description of this Concept

ConstraintDescription := Conditional formula of the constraint when the concept occurs in the compound concept

(1-2) Compound Entity Schema

{#Concept Label C-EntitySchema Attribute Value Pair;
StructureDescription
ExplanationDescription
Arch...}

StructureDescription := {Instance Label SET Attribute Value Pair;
AttributeSchemaReference...
EntitySchemaReference...
RelationSchemaReference...}

EntitySchemaReference... := EntitySchemaReference
EntitySchemaReference...
| Nil

EntitySchemaReference :=
{Instance Label Entity ref=#E-SchemaConcept Label
Attribute Value Pair} |
EntityConditionalExpression

RelationSchemaReference... := RelationSchemaReference
RelationSchemaReference...
| Nil

RelationSchemaReference :=
{Instance Label Relation ref=#R-SchemaConcept Label
Attribute Value Pair} |
RelationConditionalExpression

(2) RelationSchema

(2-1) Elemental Relation Schema

{#Concept Label E-RelationSchema Attribute Value Pair;
StructureDescription
ExpalnationDescription
{Instance Label 1 Label ref=# Concept Label from;}
{Instance Label 2 Label ref=# Concept Label to;}
[Instance Label 1 from # Concept Label]

$\{ \# \underline{\text{Concept Label}} \text{ to } \underline{\text{Instance Label 2}} \}$

$\underline{\text{StructureDescription}} := \{ \underline{\text{Instance Label}} \text{ SET } \underline{\text{Attribute Value Pair}}; \underline{\text{AttributeSchemaReference}} \dots \}$

$\underline{\text{LabelReference}} := \{ \underline{\text{Instance Label}} \text{ Label ref} = \# \underline{\text{Concept Label}} \underline{\text{Attribute Value Pair}} \}$

(2-2) Compound Relation Schema

$\{ \# \underline{\text{Concept Label}} \text{ C-RelationSchema } \underline{\text{Attribute Value Pair}};$

$\underline{\text{StructureDescription}}$

$\underline{\text{ExpalnationDescription}}$

$\underline{\text{Arch}} \dots$

$\{ \# \text{From Label ref} = \# \underline{\text{Concept Label from}}; \}$

$\{ \# \text{To Label ref} = \# \underline{\text{Concept Label to}}; \}$

$\{ \# \text{From from } \underline{\text{Instance Label X}} \}$

$\{ \underline{\text{Instance Label Y}} \text{ to } \# \text{To} \}$

$\underline{\text{StructureDescription}} := \{ \underline{\text{Instance Label}} \text{ SET } \underline{\text{Attribute Value Pair}}; \underline{\text{AttributeSchemaReference}} \dots; \underline{\text{EntitySchemaReference}} \dots; \underline{\text{RelationSchemaReference}} \dots \}$

$\underline{\text{EntitySchemaReference}} \dots :=$

$\underline{\text{EntitySchemaReference}} \underline{\text{EntitySchemaReference}} \dots$

| Nil

$\underline{\text{EntitySchemaReference}} :=$

$\{ \underline{\text{Instance Label}} \text{ Entity ref} = \# \underline{\text{E-Schema Concept Label}}$

$\underline{\text{Attribute Value Pair}} \}$ |

$\underline{\text{EntityConditionalExpression}}$

$\underline{\text{RelationSchemaReference}} \dots := \underline{\text{RelationSchemaReference}}$

$\underline{\text{RelationSchemaReference}} \dots$

| Nil

$\underline{\text{RelationSchemaReference}} :=$

$\{ \underline{\text{Instance Label}} \text{ Relation ref} = \# \underline{\text{R-Schema Concept Label}}$

$\underline{\text{Attribute Value Pair}} \}$ |

$\underline{\text{RelationConditionalExpression}}$

(3) AttributeSchema

{#Concept Label AttributeSchema value=AttributeValueList
default=AttributeValue; ExpalnationDescription}

4. Translation between CDL, RDF and XML

(1) Linguistic models of XML, RDF and CDL

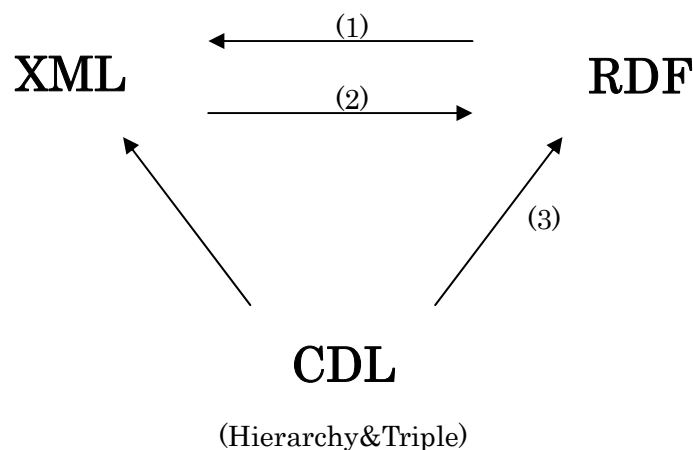
The syntactic features of CDL, RDF and XML suggest how to translate between each language.

XML defines Element tags by XML schema. Element specifies sub-elements which means specification of hierarchical structure, tree structure or nested structure.

RDF has triple model consisting of subject, property, and object. It is graphically represented by binary relation.

CDL has both hierarchical structure and triple model. Compound concept and sub-concept are super-sub relation. This structure is similar with XML hierarchy. And compound concept may include triple entity as body.

By these comparisons of linguistic models it may be concluded that CDL includes RDF and XML



(2) Translation method

The syntactic features of CDL, RDF and XML suggest how to translate between each language. Translation method is explained by the following diagram.

(2-1) From RDF to XML

This translation is directed from triple to hierarchy type. W3C specifies RDF translation to XML as RDF/XML. Basic method is to define subject, property and object as element of XML and to make the property subclass of the subject and the object subclass of the property.

Ex.

Father – named -> John (Father named John.)

→

```
<Father>
  <named>
    John
  </named>
</Father>
```

(2-2) From XML to RDF

This translation is directed from hierarchy to triple.

Ex.

```
<element1>
  <element2>
    <element3>
      xxxx
    </element3>
  </element2>
</element1>
```

→

element1 – superclass ->element2

element2 – superclass ->element3

element3 – body ->xxx

New properties “superclass” and “body” will be defined in the RDF Schema.

(2-3) From CDL to RDF

The translation is splitted into hierarchy part and triple part. The hierarchy part is translated as method (2) and the triple part is translated by using blank node of RDF.

Ex

```
{#A;  
  #1 B  
  #2 C  
  #3 D  
  [#1-#2->#3]  
}
```

→

```
#A superclass #blanc_1.  
#blanc_1 rdfs:subject #1.  
#blanc_1 rdfs:property #2.  
#blanc_1 rdfs:object #3.
```

Acknowledge

This research is supported by the Ministry of Internal Affairs and Communications of Japan under the Strategic Information and Communications R&D Promotion Programme (SCOPE).

