Grammar of CONEX Structure Model [1]

Jingzhi Guo

The previous subsection described the elements of the CONEX structure model. This subsection describes the grammar of the CONEX structure model to summarise the ideas of structure, concept, context and map that were discussed. The syntax of the CONEX grammar is largely inspired by Gulog [1]. The advantages of Gulog are the separation of schema declaration from data definition and the methods overridden on specific instances of subclasses. These provide an opportunity for the CONEX approach to separate structure from concept and to reconstruct denotation-connotation relationships in an inheritance hierarchy.

Rather than predefining the actual semantics of structures (i.e. concepts) through instantiating structures of every component EPC without interaction with other component EPCs (e.g. traditional ontologies [2] or vocabularies [3][4] in disparate EPCs), the "concept" of the CONEX approach is introduced as a result of collaborative design. The instantiation of a structure into a concept must accompany collaboration between participants of component EPCs, or at least each participant must agree that the instantiated structure has captured his/her actual meaning. The significance of instantiating structure by collaboration is that concept heterogeneity between component EPCs is eliminated. Because of the introduction of collaboration in structure instantiation, the CONEX grammar described in this subsection only includes a set of declarations that are not related to any concept semantics.

CONEX grammar defines an *alphabet* as consisting of (1) a set of *structure symbols* that are divided into symbols representing primitive structures, elementary structures and construct structures: each of which is notated by S_P, S_E, and S_C; (2) a set of concept structure symbols; (3) a set of classifier symbols; (4) a set of map symbols; (5) a set of function symbols; (6) an infinite set of variables; (7) a set of constants; (8) auxiliary symbols such as ( ), :, ::, ←, →, ↔, etc.; and finally; (9) A *term* is either a constant, a variable, or a token $s(t_1, \ldots, t_n)$ where s is a structure symbol and $t_1, \ldots, t_n$ are terms. Within a classifier, the occurrence of a concept structure is constrained by (* | + | ?) consistent with XML conformable EBNF notation **Error! Reference source not found.**, which defines the combination of concept structures.

Definition 5-6: Structures

- If s and s' are structure symbols, then s :: s' is a *structure declaration*. It is said to be that s is *substructure of* s', and conversely s' is a superstructure of s. For any structure symbol s" such that s' :: s", s is also a substructure of s".

[1] Reference for citation: "Jingzhi Guo, Grammar of CONEX Structure, Section 5.2.2 of Integrating Ad Hoc Electronic Product Catalogues through Collaborative Maintenance of Semantic Consistency, PhD Thesis, Griffith University, Australia, 2004".

- If c is an *n*-ary concept structure symbol, s_1, …, s_n are elementary structure symbols, then c(s_1, …, s_n) is a *concept structure declaration*. It is said to be that the signature of *concept structure* c is $s_1\times \ldots\times s_n$.
- If *f* is an *n*-ary function symbol, and s_1, …, s_n are structure symbols, then *f*(s_1, …, s_n) is a *function declaration*. It is said to be that the *signature* of function *f* is $s_1\times \ldots\times s_n$.
- If κ is a classifier symbol, λ_1^1, …, $\lambda_{1\ldots i}^n$ are concept structure symbols, then $\kappa(\lambda_1^1(\ldots(\ldots, \lambda_{1.i\ldots i}^n)))$ is a *classifier declaration*. It is said to be that the signature of *classifier* κ is $\lambda_1^1\times \ldots \times\lambda_{1.i\ldots i}^n$, and this signature is a *classification pattern* of classifier κ.

A structure declaration notates an abstract inheritance relation, which provides a new representation structure *modelled on* its superstructure. Concept structure declaration is a syntactic definition of a concept model, which governs how meaning should be conveyed. A function is a special structure. Its function symbol can be used as an elementary structure symbol in a concept structure declaration and thus to provide operations for concept structures. A classifier structure declaration defines a classification schema for a PRODUCT MAP (often a component EPC).

Definition 5-7: Concepts

- If *t* is a term and *s* is a concept structure symbol, then *t* : *s* is a *concept declaration*. It is said to be that *t* is a *conceptualisation* of structure *s*. If *s'* is a superstructure of *s*, then *t* is also a *conceptualisation* of structure *s'*.
- If *t* is a *leaf concept* symbol and *v* is a primitive structure symbol, then *v*←*t* or *t*→*v* is a *reification declaration*. It is said to be that leaf concept *t* has a *concept value v*.

Syntactically, concept declaration is similar to object declaration [1]. Nevertheless, semantically, concept declaration must be immediately followed by concept implementation through collaborative concept design between collaborative parties, otherwise semantic conflicts will occur between participating component EPCs. In addition, distinction should be made between conceptualisation and reification. For example, the "colour" of a refrigerator could be a substructure, which is conceptualised and reified in the way of "*red*←*c*(1.52.14.14.1-1, colour) : *c*(iid, annotation) :: *c*(IID, AN)" or "红色 ← *c*(1.52.14.14.1-1, 颜色) : *c*(标识, 注解) :: *c*(IID, AN)".

Definition 5-8: Contexts

- If s_1, …, s_n is a set of concept structures and *x* is a context symbol, then *x*{s_1, …, s_n} is a *structure context declaration*. It is said to be that concept structures have context *x*.
- If t_1, …, t_n is a set of concepts and *x* is a context symbol, then *x*{t_1, …, t_n} is a *concept context declaration*. It is said to be that concepts have context *x*.
- If v_1, …, v_n is a set of concept values and *x* is a context symbol, then *x*{v_1, …, v_n} is a *reification context declaration*. It is said to be that reified concepts have context *x*.

A given context expresses a specific semantic space. Different contexts have heterogeneous structures, concepts and concept values, which are the source of semantic conflicts. Heterogeneity can be further specified as follows:

- A set of concept structures adopts different classifiers.
- A set of concept structures with the same classifier has different conceptualisations.
- A set of concepts with the same classifier and conceptualisation has different concept reifications.

Definition 5-9: Maps

- If Ξ is a *map* symbol, ε_1 and ε_2 are *concept structure symbols*, and x_1 and x_2 are context symbols, then $\Xi(\varepsilon_1(x_1), \varepsilon_2(x_2))$ is a *one-to-one mapping declaration*. It is said to be that map Ξ has the signature of $\varepsilon_1 \leftrightarrow \varepsilon_2$.
- If Ξ is a *map* symbol, ε and $\varepsilon_1, \ldots, \varepsilon_2$ are *concept structure symbols*, and x_1 and x_2 are context symbols, then $\Xi(\varepsilon(x_1), (\varepsilon_1 \ldots, \varepsilon_n)(x_2))$ is a *one-to-many mapping declaration*. It is said to be that map Ξ has the signature of $\varepsilon \leftrightarrow \varepsilon_1 \times \ldots \times \varepsilon_n$.

A map is a special construct for linking heterogeneous contexts, which allows that concepts between different contexts are substitutable regardless of their heterogeneous classifiers, conceptualisation and reification methods.

Reference

[1] Dobbie, G. and R. Topor, "On the Declarative and Procedural Semantics of Deductive Objected-Oriented Systems", *Journal of Intelligent Information Systems*, Vol. 4, No. 2, 1995, pp. 193-219.

[2] Keller, A. M., "Multivendor Catalogues: Smart Catalogues and Virtual Catalogues", *EDI Forum: Journal of Electronic Commerce* 9(3), 1996, pp. 87-93.

[3] eCl@ss, www.eclass.de.

[4] HL7 Reference Information Model, http://www.hl7.org.