A FORMAL APPROACH TO THE SEMANTICS OF A FRAME DATA MODEL

Ulrich Reimer / Udo Hahn

Universitaet Konstanz
Informationswissenschaft
Postfach 5560
D-7750 Konstanz 1, W. Germany

ABSTRACT

Standard knowledge representation languages are seriously lacking an explicit formal semantic specification. This may cause considerable trouble when applied to large amounts of rapidly changing data.

Based on an abstract data type view of knowledge representation languages a formal definition of a frame data model is presented in terms of a denotational semantics approach using a subset of META-IV. After introducing some basic concepts of the model several semantic integrity constraints are outlined which ultimately lead to the formulation of a set of operations in the frame data model.

1. Introduction

Standard knowledge representation languages such as KRL (BOBROW/WINOGRAD 1977) or FRL (ROBERTS/GOLDSTEIN 1977) are almost exclusively characterized by syntactic specifications, but no attention is paid to the formal specification of their semantics. Recently, several formalisms have been devised which provide explicit semantic specifications, KL-ONE (as outlined in BRACHMAN 1979) being one of the most promising proposals. Exhaustive formal conditions on the operations on knowledge representation structures clearly lead to an abstract data type view of knowledge representation languages (WEINER/PALMER 1981).

The lack of semantic specifications in knowledge representation languages may cause severe problems (e.g. unpredictable side effects and inconsistent changes in the data base), which obviously become more and more serious when large numbers of knowledge structures are managed. We consider knowledge base systems to be realizations of formally defined data models. By formally specifying the semantics of the operations of an underlying frame data model we attempt to strictly constrain the actual behavior of a large knowledge base system which forms an essential part of TOPIC, an automatic text understanding and abstracting system (TOPIC 1983), currently under development at the information science department of the University of Constance, W. Germany.

The formal specification of our frame data model is based on a denotational semantics approach, applying a subset of the META-IV-Language (e.g. as illustrated in BJORNER 1980). After introducing basic concepts of our model (section 2), we shall give some examples of semantic integrity constraints which apply properties of the previously defined concepts (section 3) and finally indicate the basic operations on objects of the data model (section 4), thus providing a sketch of what the semantics of a frame data model might look like (for an extended version of this paper see (REIMER/HAHN 1983)).

2. Basic Concepts of the Data Model

The notion of a frame is defined by a mapping

FRAMES = Frame —> SLOTS

Frame denotes the set of frame identifiers and SLOTS denotes the set of slots. The notion of a slot is given by the mapping

SLOTS = Sname —> SENTRY

Sname denotes the domain of slot identifiers and SENTRY denotes the domain of actual and permitted slot entries. The actual and permitted entries for a slot are given by the mapping

SENTRY = Type —> Entries

The set Type consists of act and perm. The element act is mapped to the set of actual entries while perm is mapped to the set of permitted entries, both sets constituting the set Entries.

A network of relationally connected frames may be constructed from the relations given in the data model. We now consider single elements of the mapping domains defined above.

This work was supported by Bundesministerium fuer Forschung und Technologie / Gesellschaft fuer Information und Dokumentation under grant no. PT 200.08.

$$\begin{aligned}\text{Is-a} = \{\ &\langle f,f'\rangle \mid f, f' \in \underline{\text{dom}}\ \text{FRAMES} \wedge \\ &\wedge \sim \exists f'' \in \underline{\text{dom}}\ \text{FRAMES}: [\ \neg \langle f,f''\rangle \in \text{Is-a} \vee \\ &\qquad \vee \langle f'',f\rangle \in \text{Is-a}\ ] \wedge \\ &\qquad \wedge f'' \in \underline{\text{dom}}\ (\text{FRAMES}(f))\ ] \wedge \\ &\wedge \underline{\text{dom}}\ (\text{FRAMES}(f)) \supseteq \underline{\text{dom}}\ (\text{FRAMES}(f')) \wedge \\ &\wedge \forall s \in \underline{\text{dom}}\ (\text{FRAMES}(f)) \cap \underline{\text{dom}}\ (\text{FRAMES}(f')): \\ &\quad \text{FRAMES}(f)(s)(\underline{\text{perm}}) \subseteq \text{FRAMES}(f')(s)(\underline{\text{perm}})\ \}\end{aligned}$$

The Is-a hierarchy requires that a hyponym frame have at least the same slot identifiers as its superordinate frames and maximally the same set of permitted slot entries for corresponding slots of its superordinate frames.

A frame which is element of an Is-a tuple is called prototype:

$$\begin{aligned}&f \in \underline{\text{dom}}\ \text{FRAMES is prototype} :\Leftrightarrow \\ &\quad \exists f' \in \underline{\text{dom}}\ \text{FRAMES}: (\ \langle f,f'\rangle \in \text{Is-a} \vee \\ &\qquad \vee \langle f',f\rangle \in \text{Is-a}\ )\end{aligned}$$

A prototype f defines an equivalence class of frames:

$$\begin{aligned}ec_f := \{\ &f' \mid f' \in \underline{\text{dom}}\ \text{FRAMES} \wedge \\ &\wedge\ \underline{\text{dom}}\ (\text{FRAMES}(f')) = \underline{\text{dom}}\ (\text{FRAMES}(f)) \wedge \\ &\wedge \forall s \in \underline{\text{dom}}\ (\text{FRAMES}(f')): \\ &\quad \text{FRAMES}(f')(s)(\underline{\text{perm}}) = \text{FRAMES}(f)(s)(\underline{\text{perm}})\ \}\end{aligned}$$

The elements in ec_f are called instances of f. The equivalence class of a prototype frame consists of instance frames whose slot identifiers and sets of permitted entries are the same as those of the prototype. Only the actual slot entries may differ among the frames in the equivalence class.

The relation E-is-a (extended is-a) uses the concept of equivalence classes:

$$\begin{aligned}\text{E-is-a} := \{\ &\langle f,f'\rangle \mid f, f' \in \underline{\text{dom}}\ \text{FRAMES} \wedge \\ &\wedge [\ \langle f,f'\rangle \in \text{Is-a} \vee \\ &\quad \vee \exists f'' \in \underline{\text{dom}}\ \text{FRAMES}: \\ &\qquad [\ \langle f'',f'\rangle \in \text{Is-a} \wedge f \in ec_{f''}\ ]\ ]\ \}\end{aligned}$$

$$\begin{aligned}\text{Parts} := \{\ &\langle f,f'\rangle \mid f, f' \in \underline{\text{dom}}\ \text{FRAMES} \wedge \\ &\wedge [\ f \in \underline{\text{dom}}\ (\text{FRAMES}(f')) \vee \\ &\quad \vee \exists s \in \underline{\text{dom}}\ (\text{FRAMES}(f')): \\ &\qquad f \in \text{FRAMES}(f')(s)(\underline{\text{act}})\ ] \wedge \\ &\wedge \exists f'' \in \underline{\text{dom}}\ \text{FRAMES}: \langle f,f''\rangle, \langle f',f''\rangle \in \text{E-is-a}\ \}\end{aligned}$$

The Parts relation holds for two frames if the identifier of one frame is a slot identifier of the other or is a slot entry of the other frame and both frames have a common superordinate frame in the Is-a hierarchy.

3. Semantic Integrity Constraints

The properties of the relations defined above not only put severe restrictions on the possible relational connection between two frames, but require the existence of a relational tuple, if two frames share appropriate properties. Further restrictions are included in the data model in terms of explicit semantic integrity constraints (viewing the properties of the relations as implicit semantic integrity constraints).

As an example the following constraint holds for instances:

$$\begin{aligned}(1)\ &\forall f \in \underline{\text{dom}}\ \text{FRAMES}: \forall f' \in ec_f: \\ &\quad \neg \exists f'' \in \underline{\text{dom}}\ \text{FRAMES}: (\ \langle f',f''\rangle \in \text{Is-a} \vee \\ &\qquad \vee \langle f'',f'\rangle \in \text{Is-a}\ )\end{aligned}$$

Distinguishing two types of slots allows for a more stringent control of slot filling:

1. The set of non-terminal slots is given by
   $\text{NTSLOTS} = \{\ s \dashrightarrow \text{SENTRY} \mid s \in \underline{\text{dom}}\ \text{FRAMES}\ \} \cap \text{SLOTS}$

2. All other slots are called terminal slots
   $\text{TSLOTS} = \text{SLOTS} \setminus \text{NTSLOTS}$

The mapping SENTRY gives for each slot the set of actual and the set of permitted slot entries. The permitted slot entries for terminal slots must be specified by the user of the data model, the permitted slot entries for non-terminal slots cannot be defined externally, but are given by the following model-dependent integrity constraint which states that permitted slot entries of slot s are hyponyms of the frame with the identifier s with respect to the E-is-a relation:

$$\begin{aligned}(2)\ &f \in \underline{\text{dom}}\ \text{FRAMES} \wedge \\ &\quad \wedge s \in \underline{\text{dom}}\ \text{NTSLOTS} \cap \underline{\text{dom}}\ (\text{FRAMES}(f)) \Rightarrow \\ &\qquad \text{FRAMES}(f)(s)(\underline{\text{perm}}) := \\ &\qquad\quad \{\ f' \mid f' \in \underline{\text{dom}}\ \text{FRAMES} \wedge \langle f',s\rangle \in \text{E-is-a}\ \}\end{aligned}$$

The following constraint states that a prototype may not have a slot entry:

$$(3)\ f \text{ is prototype} \Rightarrow \forall s \in \underline{\text{dom}}\ (\text{FRAMES}(f)): \text{FRAMES}(f)(s)(\underline{\text{act}}) = \emptyset$$

4. Basic Operations in the Data Model:

In the following the basic operations are given (to simplify the notation variables are not prefixed with c in order to get their contents, as is required by META-IV).

1. Get all frame identifiers:
   $\underline{\text{dom}}\ \text{FRAMES}$

2. Get all slot identifiers:
   $\underline{\text{dom}}\ (\text{FRAMES}(f))$

3. Get all permitted slot entries:
   $\text{FRAMES}(f)(s)(\underline{\text{perm}})$

4. Get all actual slot entries:
   $\text{FRAMES}(f)(s)(\underline{\text{act}})$

5. Add a frame:
   $\text{FRAMES} \cup [\ f \dashrightarrow []\ ]$

6. Add a slot:
   $\text{FRAMES}(f) \cup [\ s \dashrightarrow [\ \underline{\text{act}} \dashrightarrow \emptyset, \underline{\text{perm}} \dashrightarrow \emptyset\ ]\ ]$

7. Add an element to the set of permitted slot entries:
   $s \in \text{TSLOTS} \Rightarrow \text{FRAMES}(f)(s)(\underline{\text{perm}}) \cup \{pe\}$

8. Fill a slot:
$\exists f' \in \underline{dom}\ FRAMES: f \in ec_{f'} \wedge$
$\wedge\ e \in FRAMES(f)(s)(\underline{perm}) \Rightarrow$
$FRAMES(f)(s)(\underline{act}) \cup \{e\}$

9. Delete a frame:
sfunc := FRAMES(f)
FRAMES \ [f --> sfunc]

10. Delete a slot:
efunc := FRAMES(f)(s)
FRAMES(f) \ [s --> efunc]

11. Delete an element of the set of permitted slot entries:
$s \in TSLOTS \Rightarrow FRAMES(f)(s)(\underline{perm}) \setminus \{pe\}$

12. Delete a slot entry:
$FRAMES(f)(s)(\underline{act}) \setminus \{e\}$

The existence of a relational edge between two frames depends only on the existence of two frames with the properties required by the relation. Therefore operations to insert or delete relation tuples need not be defined. Applying the definitional properties of the Is-a and Parts relation and the properties of the equivalence class for each prototype, the semantically correct insertion and deletion of relation tuples and instance frames will be controlled and executed by the knowledge base system which realizes the frame data model.

5. Conclusion

We have outlined some basic ideas for the formal definition of a frame data model by applying a denotational semantics approach. Next, emphasis will be given to an extension of world-dependent and model-dependent integrity constraints.

Subsequently an axiomatic specification of the frame data model will be developed. After a comparison of both approaches, further refinement of the frame data model will be based on the most appropriate approach.

REFERENCES

[1] BJORNER, D.: Formalization of Data Base Models. In: D. Bjorner (ed): Abstract Software Specifications. Berlin: Springer, 1980, pp.144-215.

[2] BOBROW, D.G. / WINOGRAD, T.: An Overview of KRL, a Knowledge Representation Language. In: Cognitive Science 1. 1977, pp.3-46.

[3] BRACHMANN, R.J.: On the Epistemological Status of Semantic Networks. In: N.V. Findler (ed): Associative Networks. New York: Academic Pr., 1979, pp. 3-50.

[4] REIMER, U. / HAHN, U.: A Formal Approach to the Semantics of a Frame Data Model. Konstanz: Universitaet Konstanz, Informationswissenschaft, 1983 (TOPIC-3/83).

[5] ROBERTS, R.B. / GOLDSTEIN, I.P.: The FRL Manual. MIT AI-Lab., 1977.

[6] TOPIC: A System for Automatic Text Condensation. In: ACM SIGART Newsletter #83, 1983. pp. 20-21.

[7] WEINER, J.L. / PALMER, M.: The Design of a System for Designing Knowledge Representation Systems. In: Proceedings IJCAI-81, 1981, pp.277-282.