INFORMATICA, 1997, Vol. 8, No. 1, 139–152

FLEXIBLE CLASSIFICATION IN ProObj

Dieter DODENHÖFT, Georg STROBL, Andreas STRASSER

Technische Universität München, Institut für Informatik
Arcisstr. 21, D–80290 München, Germany
E-mail: strobl@informatik.tu-muenchen.de

Abstract. ProObj is a Prolog based system for knowledge representation which was strongly influenced by object-oriented and frame-based systems. The paper shortly describes ProObj and then presents a classification mechanism which is based on the ideas of classifiers in KL–ONE like systems.

As a new and very flexible feature we present a user-directed control of classification process. The ProObj classifier gives the user the possibility to guide the classification process by excluding attributes and facets – elements of our representation formalism – from being considered in the classification. By this mechanism we gain a substantial improvement of the efficiency of the classification process. Furthermore, it allows a more flexible and adequate modelling of a knowledge domain. It is possible to build a knowledge base under a particular view where only those attributes of concepts are considered for classification which seem to be relevant for the structure of the domain hierarchy.

Key words: knowledge representation, object-oriented systems, frame-based systems, logic programing, classification.

1. Introduction. Despite a variety of existing tools for knowledge representation still many new systems are developed which try to improve some aspects or features of the known paradigms.

ProObj (Fischer and Lippert, 1988; Dodenhöft and Strobl, 1988) is a system for knowledge representation in the tradition of object-oriented and frame-based systems, which was developed at the Technical University of Munich. It was implemented in a Prolog environment and its main objective is to integrate already known ideas from object-oriented and frame-based systems, to extend some of the aspects and to combine them with the advantages of Prolog.

With the formalism provided by ProObj it is especially easy to represent terminological knowledge, i.e., taxonomies of descriptions of a particular do-

main. As stated in Lipkis (1982) taxonomies are one of the most natural and useful ways to organize descriptive terms in a knowledge base. Therefore it is important to automatically detect the correct position of a description within the taxonomy, i.e., to provide a classifier.

In the following we will give a brief introduction into the ProObj system. Section 3 will then describe the classifier of ProObj which is different in some ways to the "classical" KL–ONE like classifiers due to the differences of the underlying representational approaches. The ProObj classifier also introduces a new feature which permits a user-directed control of the classification process, e.g., to enhance its performance – a crucial problem with most classifiers. Finally in Section 4 we will conclude our presentation with some examples showing the advantages of our flexible classification in ProObj.

2. ProObj – a short overview. In this Section we will briefly describe the formalism for knowledge representation and point out the main features of ProObj.

The ProObj system provides 4 main elements for structuring the domain knowledge and building up a knowledge base: classes, attributes and facets (which can be viewed as the descriptional part of the system) and instances (which make up the assertional part of ProObj).

The basic idea in ProObj is to structure knowledge with regard to conceptual entities and to unite similar information into **classes**. Each class describes a set of individuals. Thus, in our notion a class consists of a set of attributes and describes a set of instances.

Attributes in turn are complex structures which describe properties or aspects that are characteristic for a class of individuals. Attributes consist of a set of facets. The notion corresponds to the notion of "slots" in frame-based systems (compare Roberts and Goldstein (1977) or Goldstein and Roberts (1979)). In addition ProObj distinguishes 3 types of attributes regarding the meaning of the attributes for the class description and the intention upon their creation.

- **Maintenance attributes** provide statements which concern the class itself, e.g., the relation to other classes or the number of actually existing instances belonging to that class.
- **Class attributes** indicate universal properties, i.e., properties which are identical for all elements of a class. Similar to class variables in object-oriented languages their values are valid for all individuals of a class.

- **Instance atributes** provide statements that describe properties which are typical but have in general different values for every element of a class.

Finally, the most interesting part of ProObj are the **facets**. Like in frame-based systems a facet is a pair, which consists of a facet name and a facet value. Facets provide additional information for the attribute values (e.g., type information, restrictions or dependencies from other values, units of measure, default values, attached procedural knowledge and many more) so the system can perform some consistency checks and already stored attribute values have a detailed description. However, to state these intuitive semantics of facets in a formal and declarative way ProObj uses some additional mechanism.

The semantics of each facet name is laid down by a Prolog predicate. So, when a new instance is created or an attribute value is changed the new attribute value is checked against all facet definitions for that attribute using the corresponding Prolog predicates. Especially for facets which restrict the possible attribute values this method is quite straight forward. But, as these Prolog predicates can be defined by the user without any restrictions using full Prolog, one can also specify the semantics of, e.g., procedural attachment or demons. In order to help the user in his task of knowledge representation ProObj provides some predefined facets, but as the system should be as flexible as possible the user can specify new facets or can change the semantics of the predefined ones. For each new facet the user must provide an appropriate Prolog predicate[1]. This predicate must be fulfilled each time the user wants to set or change the corresponding attribute values.

In Fig. 1 an example is given how the semantics for a facet *range_restriction* can be defined by a Prolog predicate. A value for an attribute which contains the facet *range_restriction* should be only accepted if that value lies within the bounds of an interval specified by the value of a *range_restriction* facet. This predicate is tested every time a new attribute value is assigned.

As already mentioned each **instance** belongs to a class which holds additional information about its structure and semantics. An instance holds assertional information and consists of attribute values for the attributes specified in the class definition. To be more efficient only the values for instance attributes are stored in the instances themselves. As attribute values arbitrary Prolog

[1] Note that the full specification of the semantics of a facet must also contain subsumption relation for the facet values (see Fig. 2).

```
facet (range_restriction, Facet_value, Attr_value) : –
        Facet_value = [Lower_bound, Upper_bound],
        Attr_value ⩾ Lower_bound,
        Attr_value ⩽ Upper_bound.
```

Fig. 1. Definition for the facet *range_restriction*.

terms are allowed which satisfy the facet definitions. Especially *complex Prolog structures* and *references to other instances* are possible.

One important feature of the ProObj system is the inheritance mechanism, which allows hierarchical as well as multiple inheritance. The inheritance strategy which it employs is a depth-first left to right search for attribute definitions in the superclasses of a class. The inheritance is calculated immediately after every class definition or modification.

Apart from changing the inherited description of a concept by redefining inherited attributes or adding new ones, ProObj provides the possibility to refine even an inherited attribute description. Subclasses can add new or redefine existing facets for inherited attributes, so that the description of these attributes gets more specific. Additionally the ProObj system provides a mechanism for preserving the consistency of a knowledge base when redefining existing facets.

Take as an example the facet *range_restriction* which restricts attribute values to an interval of possible values. If that facet is redefined it will be sensible that the new range should be within the bounds of an inherited *range_restriction*. In the same way this range should be more general than corresponding *range_restrictions* in subclasses. However, as the user has the possibility to define arbitrary facets, he must also provide explicity the information for this specialization-relation. This is again done by a Prolog predicate. The following `subsumes` predicate in Fig. 2 defines such a relation.

The ProObj interface provides operations for creating and retrieving its fundamental elements. But apart from that also operations exist for modifying or deleting classes, as well as attributes, facets or instances. It is also possible to reorganize a class hierarchy by linking individual classes to other classes (inheritance link), or by deleting such a link. All these operations are implemented as Prolog predicates and upon backtracking retrieval operations return alternative solutions.

subsumes (range_restriction, Gen_facet_value, Spec_facet_value) : –

Gen_facet_value = [Gen_lb, Gen_ub],

Spec_facet_value = [Spec_lb, Spec_ub],

Gen_lb $\leqslant$ Spec_lb,

Gen_ub $\geqslant$ Spec_ub.

Fig. 2. Definition of the subsumption relation for the *range_restriction* facet.

3. The ProObj classifier. As mentioned in the previous section, ProObj is a system for representing terminological knowledge, i.e., taxonomies of concept descriptions. In such systems – KL–ONE like systems are the most known representatives of this category – it is of extreme importance to have means for automatically detecting the correct position of a description within the taxonomy. The process of determining the taxonomic relationship of a new description with already existing descriptions in the taxonomy and the incorporation of this relationship into the knowledge base is commonly referred to as *classification.*

In the following we will present the ProObj classifier and describe its distinctions to the "classical" KL–ONE like classifiers. The two most interesting features of this classifier are (i) the definition of the subsumption relation, i.e., the semantics of the concept hierarchy are fully under control of the user and (ii) attributes and facets can be excluded from classification in order to improve efficiency or to allow a more flexible modelling of a knowledge domain. In addition, it is possible to check the consistency of manually created concept hierarchies.

3.1. The ProObj concept hierarchy. The position of a new concept (i.e., class) in a ProObj hierarchy depends on the subsumption relationships of this new concept to already existing ones in the hierarchy. These subsumption relationships are represented by so called "superconcept links" (or "inheritance links") which can be established in two different ways:

(i) manually by the knowledge engineer who determines the position of the concept within the hierarchy corresponding to his/her's intention or

(ii) automatically by the classifier which has to detect the right place in the hierarchy corresponding to the description of the new concept.

Like in KL–ONE we can also distinguish between two different meanings of the superconcept link. If all properties which can be inherited from the superconcept are represented explicitly on the subconcept the superconcept link does not affect the definition of the subconcept. It only states that there exists a subsumption relationship between the two concepts. Otherwise, if there exist properties on the subconcept only because they are inherited from the superconcept, the superconcept link is definitional – it is a means for defining the subconcept. A concept in a frame-based representation paradigm does not only consist of subparts which have already been defined in the hierarchy, it's constituents may also be locally defined attributes. This means that possible a new concept has no initial location in the hierarchy. Therefore, due to ProObj's dynamic inheritance mechanism a new concept either has to be defined completely before classification or a partially defined concept has to be put into the hierarchy manually and the inheritance process be invoked before a futher classification of such a "completed" concept description can proceed. "Further classification" means that it is possible, that a first manual integration of a new concept into the hierarchy may not have expressed all possible subsumption relationships of this concept or the given relationships have been incorrect.

3.2. Definition of the subsumption relation. The underlying representational paradigm of ProObj is that of *frames* and *objects*. Therefore, refinement of a concept description is primarily based on two different methods:

(i) addition of attributes;
a class which has all of the attributes of another class and at least one additional attribute is considered to be more specific than this one;

(ii) refinement of attributes;
if a new description of a concept is introduced which consists of the same attributes as the former one but whose attribute descriptions – the facets – have been refined, then this is also considered to be a more specific description. Refinement of facets occurs when additional facets are introduced to the attribute description or existing facets are redefined with a more specific semantics. For an example of a facet refinement see Section 2.

Refining a concept description, i.e., creating a new concept which is more specific than the already existing one and putting it into the concept hierarchy implies of course establishing a subsumption relation between these two con-

cepts. In this context it is essential to understand that the subsumption relation of two concepts is defined exclusively upon the subsumption relation of the corresponding attributes of the concept description which in turn depend on the subsumption relation of the facets they consist of. The subsumption of facets has to be defined by the user when a new facet is introduced to the system. Therefore, the subsumption relation of concepts in ProObj is fully under control of the builder of a knowledge base and correspondingly he/she is fully responsible for a proper semantics and adequacy of the concept hierarchy.

Contrary to, e.g., semantic network the correspondence of components of concept descriptions (i.e., the attributes of the concepts) in our representational paradigm is simply expressed by the names of the attributes – attributes of two concepts which have the same name are considered to bear the same "meaning" for their concepts.

Consequently, the ProObj classifier reflects this view of concept refinement and, therefore, the subsumption conditions in ProObj can be defined as follows (see also Lippert (1989)):

A concept **A** subsumes a concept **B** if:

- the description of B contains at least all the attributes of A or attributes of A which are not contained in the description of B are excluded from classification;
- all attributes of B contain at least all facets of the corresponding attributes of A or facets of attributes of A which are not contained in the corresponding attributes of B are excluded from classification;
- all facets of A subsume the corresponding facets of B except those which are excluded from classification; subsumption of facets refers to proving the user defined `subsumes` predicate for those facets.

In order to give a formal definition of subsumption in ProObj, we have to introduce the following definitions:

Let X be a class (concept) description.

$Attr(X)$	denotes the set of attributes of X
$Attr\_name(X_i)$	denotes the name of the attribute X_i, $X_i \in Attr(X)$
$Attributes(X)$	denotes the set of attributes of X, without the set of maintenance attributes for storing the super-/subclasses of a class

	$Attributes(X) = Attr(X) \setminus \{X_i \in Attr(X) \mid Attr\_name(X_i) = superclasses \vee Attr\_name(X_i) = subclasses\}$
$Fac(X_i)$	denotes the set of facets of X_i, $X_i \in Attributes(X)$
$Fac\_name(X_{ij})$	denotes the name of the facet X_{ij} $X_{ij} \in Fac(X_i)$, $X_i \in Attributes(X)$
$excluded(Y)$	$Y \in Attributes(X) \vee Y \in Fac(X_i)$, $X_i \in Attributes(X)$ a predicate which holds if Y is excluded from classification

Considering these definitions, the subsumption relation between two concepts can be defined as follows:

Class A **subsumes Class** $B \iff$

$\forall A_i,\ A_i \in Attributes(A) \wedge \neg excluded(A_i),$

$\forall A_{ij},\ A_{ij} \in Fac(A_i) \wedge \neg excluded(A_{ij})$

the following conditions (1) and (2) hold:

(1) $\exists_1 B_k, B_k \in Attributes(B)$:

$\neg excluded(B_k) \wedge$

$Attr\_name(A_i) = Attr\_name(B_k)$

(2) $\exists_1 B_{kl}, B_{kl} \in Fac(B_k)$:

$\neg excluded(B_{kl}) \wedge$

$Fac\_name(A_{ij}) = Fac\_name(B_{kl}) \wedge$

$\texttt{subsumes}(A_{ij}, B_{kl})$

3.3. Exclusion from classification. An important design criteria for the ProObj classifier was to give the user the possibility for guiding the classification process. There are two possible motivations for this capability:

- to allow a more flexible and adequate modelling of a knowledge domain;
- to improve the efficiency of the classification process.

In a real application domain one has to deal with concept hierarchies of tremendous size. Usually, the concept descriptions in such hierarchies also tend to be very exhaustive and contain information which does not originally belong to the represented knowledge but is important for the maintenance of

the knowledge base. (In ProObj we provide a specialized type of attributes in order to represent this kind of information: the so called *maintenance attributes*). This implies that classification within such a hierarchy would lead to a great extent of useless work, because it doesn't yield any contribution to the structuring and modelling of the "real" knowledge. An example for this is given in Section 4.2.

Therefore, the ProObj classifier has the capability of excluding attributes as well as facets from the classification process. This means that the user has the possibility to build a knowledge base under his/her's personal view where only those attributes are considered for classification which seem to be relevant for the structure of the hierarchy.

4. Some applications. In this section we give some examples showing ProObj's flexibility with respect to classification and subsumption.

The number of existing tools for building expert systems is increasing. At the very beginning of building a knowledge-based-system it is often impossible to select the best one out of this set of tools, especially because the features of the underlaying representation paradigm are often unclear and the structure of the domain knowledge is also badly understood.

Take as an example KL–ONE systems, in which you have basically two features for describing attributes (i.e., *roles* in KL–ONE), namely *value-restriction* (`v/r`) and *number-restriction* (`n/r`). Using the `v/r` you can give a *type-information*, whereas the `n/r` restricts the cardinality of the set of possible role-fillers. KL–ONE provides only these fixed number of facets with a given predefined semantics, especially with a fixed subsumption relation. ProObj provides an easy way to specify the semantics of new facets and to integrate such new definitions of facets into the system.

4.1. Simulating other object-oriented systems. Using ProObj it is very easy to simulate different other object-oriented or frame-based knowledge-representation-systems. To show this we want to simulate (a subset of) KL–ONE. As mentioned above KL–ONE describes attributes (roles) using the `n/r` and `v/r`. Classification is based on corresponding values for these facets. A simulation in ProObj, therefore, requires the definition of these two facets and (besides the `facet`-predicates) the definition of the subsumption-relation between facet-values (compare Fig. 3).

```
subsumes (v/r, F_Value_SUPERclass, F_Value_SUBclass) : -
                is_subclass (F_Value_SUPERclass,
                             F_Value_SUBclass).

subsumes (n/r, F_Value_SUPERclass, F_Value_SUBclass) : -
                F_Value_SUPERclass = [L1,U1],
                F_Value_SUBclass = [L2,U2],
                L1 <= L2,
                U1 >= U2.
```

Fig. 3. Definition of the subsumption relation for the facets `v/r` and `n/r`.

Such simple definitions would lead to a KL–ONE-like behaviour[2] of ProObj, i.e., the consistency of the net with respect to the subsumption-relation between the given facets `v/r` and `n/r` is guaranteed.

4.2. Enhanced modelling capabilities. In the first phase of knowledge-acquisition the structure of the domain is often unclear. Are `n/r` and `v/r` sufficient and adequate for describing a certain domain? Is there any information concerning entities of the knowledge-base, which is useful for implementational aspects, but should not be included in the reasoning and classification process? Using our ProObj simulation of KL–ONE you can first try to formalize your knowledge using only the capabilities of KL–ONE. Whenever you see that there is a necessity of modifying the system you can do this very flexible. So you can do the following modifications.

- **Adding facets.** ProObj allows you to define *additional facets*, e.g., for managing default-information. This is similar to KEE (1986), but with ProObj you can additionally describe the semantics of this new facet with regard to classification. There are two possible ways to specify

[2] Clearly, there is a number of rather complex features in KL–ONE, e.g., role-value-maps, but such features are rarely implemented in existing systems. For example BACK (see Peltason et al., 1989) excludes such concepts in the current implementation. Currently we are working on a full simulation of the KL–ONE-dialect BACK.

the semantics of such a `default`-facet with respect to subsumption and classification:

- excluding the `default`-facet from classification, formally:

```
:- exclude_facet([default]);
```

- defining a subsumption-relation for the new facet. This can be done simply by adding a new `subsumes`-predicate; in the example of the default-facet it could make sense to define the subsumption-relation to hold between any default-values:

```
subsumes (default,_,_).
```

Both ways are possible, although the exclusion of this facet seems to represent the semantics of default-information more clearly.

- **Excluding attributes from classification.** Suppose you want to depict classes on the screen of your workstation using various types of *icons*. For storing this icon-type you can add a special attribute, e.g., called `icon`. Although this is not knowledge about the domain, it seems to be more clear to store such information within the corresponding class and not outside the system. But it makes no sense to take this attribute into account for the classification. Therefore, you can exclude the `icon`-attribute simply by saying:

```
:- exclude_attribute([icon]).
```

- **Extending facets.** Using the above KL–ONE-simulation it is possible to define only single interval as a number-restriction. But it is for example impossible to describe a court of justice as consisting of either two, four or five judges[3] (but not three judges), because you can only give *one single* interval as `n/r`. ProObj allows you to extend syntax and semantics of the `n/r`-facet, e.g., through using *sets of intervals*. The only thing to do is to extend the corresponding facet- and subsumption-predicate (compare Fig. 4 for the definition of the subsumption-relation). After

[3] Which could be a legal restriction.

```
subsumes (n/r, FSUPER, FSUB) : -
                FSUB = [Interval1 | Rest],
                interval_is_in(Interval1, FSUB),
                subsumes (n/r, Rest, FSUB).
```

Fig. 4. Definition of the subsumption relation for extended `n/r` facet.

such a modification it is possible to describe our court as having [[2,2], [4,5]] members.

REFERENCES

Dodenhöft, D., and G. Strobl (1988). *ProObj – Ein Prologbasiertes Objektsystem.* Internal Report for Digital Equipment International, Technische Universität München (in German).

Fischer, K., and W. Lippert (1988). *ProObj – Implementierung Eines Prologbasierten Objektsystems Mit Dynamischer Vererbung.* Technical Report, Technische Universität München (in German).

Goldstein, I., and R. Roberts (1979). Nudge, a knowledge-based scheduling program. In D. Metzing (Ed.), *Frame Conceptions and Text Understanding.* Gruyter-Verlag, Berlin, New York. pp. 26–45.

KEE (1986). *KEE Software Development System User's Manuel.* Intellicorp, Mountain View, California.

Lipkis, T. (1982). *A KL-ONE classifier.* BBN-Report, No. 4842, Bolt Beranek and Newman Inc., Cambridge, MA.

Lippert, W. (1989). *Konzeption und Entwicklung Eines Klassifikators für ein Objektorientiertes Wissensrepräsentationssystem (ProObj).* Diploma Thesis, Technische Universität München (in German).

Peltason, C., A. Schmiedel, C. Kindermann and J. Quantz (1989). *The Back System Revisited.* Technical Report KIT-Report, No. 75, Technische Universität Berlin.

Roberts, R.B., and I.P. Goldstein (1977). *The FRL Manual.* Technical Report AIM–409, AD–A052 310, Artificial Intelligence Laboratory, MIT.

Received January 1997

D. Dodenhöft studied computer science at the Technische Universität of Munich where he received his diploma degree in 1987 and his doctoral degree in 1995. From January 1988 until August 1995 he was a scientist at the Technische Universität of Munich, where he worked on knowledge representation in manufacturing environments at the artificial intelligence research group under the chair of Prof. Dr. E. Jessen and at the real-time systems and robotics group of Prof. Dr. H.-J. Siegert. His doctoral theses was on hybrid knowledge representation by a tight coupling of a rule-based and a frame-based system. Since August 1995 he is a senior software engineer at sd&m (software design and management) in Munich. His current research interests include knowledge representation and reasoning, especially for software engineering issues.

G. Strobl studied computer science at the Technische Univertität München since November 1979 and received his Diploma in May 1986. He joined the group in June 1986 and worked on object-oriented knowledge representation and knowledge-based design systems for aircraft construction and mechanical engineering. From 1987 to 1990 he was leader of the DFA project. His current research interests include modelling of technical systems, specialized knowledge representation languages, formal ontologies and reuse of knowledge bases.

A. Strasser studied computer science at the Technische Universität of Munich where he received his diploma degree in 1985 and his doctoral degree in 1991. From 1986 until 1991 he worked on expert systems for legal applications at the artificial intelligence research group at the chair of Prof. Dr. E. Jessen. His doctoral theses was on domain specific knowledge representation languages and transformation of knowledge bases. Since 1991 he is with Siemens AG in Munich where he works in the area of management, diagnosis and service of large telecommunication systems.

ProObj KALBOS LANKSTAUS KLASIFIKAVIMO PRIEMONĖS

Dieter DODENHÖFT, Georg STROBL, Andreas STRASSER

Straipsnyje trumpai apžvelgiama Prologo idėjomis grindžiama objektinė (freiminė) žinių vaizdavimo kalba ProObj ir nagrinėjamos tos kalbos lankstaus klasifikavimo priemonės. ProObj kalbos lankstaus klasifikavimo priemonės remiasi klasifikavimo idėjomis, pasiūlytomis KL-ONE šeimos kalboms. Nauja ir labai svarbi mūsų siūloma idėja yra sudaryti naudotojui galimybę pačiam valdyti klasifikavimo procesą. Mūsų siūlomi klasifikavimo būdai yra žymiai efektyvesni už anksčiau naudotus ir leidžia tiksliau modeliuoti nagrinėjamą dalykinę sritį. Be to, naudojant mūsų idėjas, galima sukonstruoti žinių bazes, kurias skirtingos naudotojų grupės gali matyti skirtingai. Tokiomis bazėmis galima modeliuoti skirtingus požiūrius į nagrinėjamą dalykinę žinių sritį.