

# Learning with Feature Description Logics

Chad M. Cumby and Dan Roth

Department of Computer Science  
University of Illinois, Urbana, IL. 61801, USA  
{cumby, danr}@uiuc.edu

**Abstract.** We present a paradigm for efficient learning and inference with relational data using propositional means. The paradigm utilizes description logics and concepts graphs in the service of learning relational models using efficient propositional learning algorithms. We introduce a *Feature Description Logic* (FDL) - a relational (frame based) language that supports efficient inference, along with a *generation function* that uses inference with descriptions in the FDL to produce features suitable for use by learning algorithms. These are used within a learning framework that is shown to learn efficiently and accurately relational representations in terms of the FDL descriptions.

The paradigm was designed to support learning in domains that are relational but where the amount of data and size of representation learned are very large; we exemplify it here, for clarity, on the classical ILP task of learning family relations. This paradigm provides a natural solution to the problem of learning and representing relational data; it extends and unifies several lines of works in KRR and Machine Learning in ways that provide hope for a coherent usage of learning and reasoning methods in large scale intelligent inference.

## 1 Introduction

In a variety of AI problems such as natural language understanding related tasks and visual inference there is a need to learn, represent and reason with respect to definitions over structured and relational data. Examples include the problem of identifying relations such as “A is the assassin of B”, when attempting to answer a free-form question given potentially relevant text, identifying a seminar’s speaker given the seminar’s announcement, detecting people in an image or defining a policy that maps states and goals to actions in a planning situation.

In these cases it is natural to represent concepts relationally; propositional representations might be too large, could lose much of the inherent domain structure and consequently might not generalize well. In recent years, this realization has renewed the interest in studying relational representations both in the knowledge representation and reasoning (KRR) community and in the learning community. The challenge is to provide the expressivity necessary to deal with large scale and highly structured domains such as natural language (NL) and visual inference and at the same time meet the strong tractability requirements for these tasks.

This challenge has been a topic of active research in the knowledge representation and reasoning community for more than a decade. The main effort has been to identify

classes of representations that are expressive enough to allow reasoning in complex situations yet are limited enough as to support reasoning efficiently [18, 28]. It has become clear that propositional representations are not sufficient, and much effort has been devoted to studying languages that are subsets of first order logic, such as description logics and frame representation systems [1, 6], as well as probabilistic augmentations of those [16].

The expressivity vs. tractability issue has been addressed also from the learning perspective, and a similar tradeoff has been observed and studied. While, in principle, Inductive Logic Programming (ILP) methods provide the natural approach to these tasks in that they allow induction over relational structures and unbounded data structures, theoretical and practical considerations render the use of unrestricted ILP methods impossible. These methods have also been augmented with the ability to handle uncertainty [14] although, as expected, this makes some of the computational issues more severe - studies in ILP suggest that unless the rule representation is severely restricted the learning problem is intractable [22, 8, 4, 5]. The main way out of these computational difficulties has been via the use of propositionalization methods that attempt to learn classifiers for relational predicates via propositional algorithms, mapping complex structures to simple features [17].

This paper develops a paradigm for efficient learning and inference with structured data by building on progress made in both communities. This paradigm utilizes models developed in the KRR community, such as description logics and concept graphs, in the service of learning relational models via efficient propositional learning algorithms.

We present a learning framework built around a *Feature Description Logic* (FDL) - a relational (frame based) language that supports efficient inference, along with an efficient *generation function* that uses inference with descriptions in the language in order to produce features suitable for use by learning algorithms.

In this paradigm, the description logic is an intermediate step, rather than the final representation, as is usual in KRR. The basic inference step, subsumption, is used as a means to transform a domain element, e.g., a natural language sentence, and represent it in terms of a richer vocabulary - descriptions in the FDL. This representation, in turn, may serve as an input to any propositional learning algorithm, including probabilistic algorithms, to yield structures in which sought after predicates are represented as functions (or conditional probabilities) over the relational descriptions.

In this respect our approach differs from standard ILP approaches and most propositionalization techniques. Features are generated up front before any learning stage in a data-driven way, allowing us to dictate the level of complexity of our intermediate representation before any search for a learned function occurs. Thus particularly expressive features that could not possibly be generated during the search itself are allowed to influence our final learned function in a significant way.

This paper provides a formal syntax and semantics for a specific feature description language (FDL). As in other description logics, we describe concepts in terms of *individuals* possessing attributes and roles in relation to other individuals. We then show the equivalence of descriptions in FDL to a class of concept graphs and use this to prove efficient subsumption between descriptions. We claim that the FDL language presented is

only one member in a family of languages and that, in fact, several existing description languages, deterministic and probabilistic, can be used within our framework.

Our main construction is a Feature Generation Function that, given a FDL description and a domain element (e.g., an image, a sentence) represented as a relational structure, uses subsumption to re-represent it using a new vocabulary, in terms of the FDL descriptions. This representation, in turn, can be used also as a feature representation, usable by general purpose propositional learning algorithms.

Finally, after describing the general learning framework and the details of the FDL, we present an application of our technique to the “classical” ILP example of learning family relationships.

The problem of learning definitions for kinship relations has been an arch-typical example of relational learning since the late 80’s/early 90’s [12, 23]. Simply put, given two members of a family, the task is to decide what their relation is. It has been argued that this task is a perfect example of a situation where relational learning is necessary, as both the concepts to be learned and all possible antecedents to a learned function are relations. In addition, ILP systems are purported to learn simple rules using a relatively small number of highly structured data instances. Thus it is an important task to test the viability of our relational learning paradigm. However, while our system will be shown to perform well on this simple, clean example, it is particularly well-suited to problem areas where concepts must be learned as a function of a mixture of ground and unground propositional and relational terms. Such problems are found extensively in natural language understanding, visual recognition, and other commonly occurring areas.

While the learning approach is presented here as an approach to learn a definition for single predicates, we view this in a wider context. Learning definitions may be used to enrich vocabulary describing the input data; the FDL can then be used incrementally to produce useful features again and subsequently to build up new representations in terms of those in a manner similar to the one envisioned in [29]. Such a system might integrate easily into a programming platform, allowing researchers to construct large scale learning-based architectures to solve complex AI problems in areas such as natural language processing. It then becomes even more crucial that the basic components of this system are articulated in a language whose structure and meaning are well understood.

## 2 The Learning Framework

In this section we introduce a learning framework that can be used to generate classifiers for relational learning problems utilizing propositional learning algorithms. The key to our propositionalization technique is the combination of a feature extraction stage composing the data in a graph-based manner to produce features, and a classification stage utilizing a feature-efficient propositional learning algorithm to learn a function for each relation.

In our framework, as in ILP, observations in the domain are mapped into some collection of predicates that hold over elements in the domain. From this the task then becomes to produce a classifier to predict which predicates hold for some particular el-

ements. For example, we may wish to predict that for some domain elements  $X$  and  $Y$ , the predicate  $father(X, Y)$  holds. To accomplish this task using standard propositional learning algorithms, we must generate examples in the form of lists of active propositions (features) for each predicate to be learned. Propositions of this form may either be fully ground as in the predicate  $father(john, jack)$ , or individually quantified as in the predicate  $\exists X father(john, X) \wedge father(X, harry)$ . Each list can also serve as a negative example for every other possible relation between elements that does not hold. These examples are used to train a classifier for each relation type, based on a linear function over the feature space. Our first major task then becomes to re-represent the data in a manner conducive to producing features for which we can learn a good discriminant function. Later we will show how by defining a focus of attention region “near” the predicate to be learned, we can differentiate the range of features produced based on the current target.

In order to facilitate this re-representation, we rely on a feature description language designed to model and work with the data in a graph-based manner. In theory and practice, this language is similar to the one developed in [7, 25].

As we will see, the feature extraction method we present operates with the so-called closed-world assumption, generating only the features judged to be active in the example relative to the generating descriptions. All other features are judged to be inactive, or false. As it may be inefficient or impossible to list all features for a particular interpretation, this is a performance boon. Thus our learning algorithm should be able to accept examples represented as variable length vectors of only positive features.

In addition, our method provides the flexibility to generate a large number of features by designating a smaller set of “types” of features, so our learning algorithm should be able to learn well in the presence of a possibly large number of irrelevant features.

For the learning component, we use the SNoW learning system. SNoW<sup>1</sup> is a multi-class propositional classifier suited to a sparse representation of feature data of variable length and uses a network of linear functions to learn the target concept. It has been shown to be especially useful for large scale NLP and IE problems [15, 25, 11]. As SNoW employs a feature-efficient learning algorithm, Winnow [19], to learn a linear function over the feature space, it learns well in the presence of many irrelevant features. The linear threshold function also makes our hypotheses more expressive, as well as easier to learn.

## 2.1 Feature Description Logic

Here we describe our basic Feature Description Logic (FDL), by providing its formal syntax and semantics. We provide example descriptions from the family relations domain similar to those used later in the actual experiments.

As in most formal description logics, FDL descriptions are defined with respect to a set  $X$  of *individuals*. However, unlike most KL-ONE-like description logics, the basic alphabet for FDL descriptions includes *attribute*, *value*, and *role* symbols. We differentiate attribute from role descriptions and our basic primitive description is an

<sup>1</sup> Available at <http://L2R.cs.uiuc.edu/~cogcomp/>

attribute-value pair. We also allow a non-functional definition of attribute descriptions and role descriptions; thus an attribute describing an individual may take many values and a role describing an individual could be filled by several different individuals.

**Definition 1.** A FDL description over the attribute alphabet  $Attr = \{a_1, \dots, a_n\}$ , the value alphabet  $Val = v_1, \dots, v_n$ , and the role alphabet  $Role = \{r_1, \dots, r_n\}$  is defined inductively as follows:

1. For an attribute symbol  $a_i$ ,  $a_i$  is a description called a sensor. For some value symbol  $v_j$ ,  $a_i(v_j)$  is also a description, called a ground sensor.
2. If  $D$  is a description and  $r_i$  is a role symbol, then  $(r_i D)$  is a role description.
3. If  $D_1, \dots, D_n$  are descriptions, then  $(\mathbf{AND} D_1, \dots, D_n)$  is a description. (The conjunction of several descriptions.)

This definition allows for the recursive construction of complex FDL descriptions such as:

*Example 1.* We can imagine that one might want to describe the set of all people who are the father of someone's father, and also someone's grandfather. They might use this description:

**(AND (father (father name)) (grandfather name))**

We note that to avoid undecidability results known for subsumption in KL-ONE-like languages with unrestricted fillers for role descriptions [26], we omit constructions from our language designed to capture equality between individuals in the extensions of different subdescriptions. Feature value maps or SAME-AS type operators as found in CLASSIC-like languages allow this type of constraint, but, in languages in which roles may take an undertermined number of fillers, usually render subsumption between descriptions undecidable. Another type of equality constraint that might better suit our purposes in extracting useful features is a constraint over the values taken for particular attributes for different individuals. For example, in a natural language understanding application, we might wish to capture the fact that the plurality of the subject matches that of the main verb. Such constraints might also prove to make subsumption between descriptions difficult.

Luckily, if the range of possible values for such attributes is known, we can simulate this constraint simply by enumerating each possible case in which the constraint is satisfied with a description. The union of all those individuals described by each is then the set of individuals that satisfy the general constraint.

*Example 2.* In our natural language task, we want to describe the set of words that act as the subject of the sentence, and for which the plurality of the subject matches that of the main verb. We describe plurality with the attribute *number*, which can take the values *single*, and *plural*. We also assume that the *subject-of* role is defined for the subject and main-verb of the sentence. The following two descriptions then capture this set of words:

**(AND number(single) (subject-of number(single)))**

**(AND number(plural) (subject-of number(plural)))**

We now turn to the semantics of FDL descriptions. This discussion follows a model-theoretic framework similar to that laid out in [6, 1]. This definition uses the notion of an interpretation [20], and that of an *interpretation function* which can be viewed as the function that encodes the information about domain. For a domain element  $z$  we denote by  $z^I$  its image under the interpretation function. Since it will be clear from the context, we use the same notation for an interpretation and an interpretation function.

**Definition 2.** *An interpretation  $I$  consists of a domain  $\Delta$ , for which there exists an interpretation function  $I$ . The domain is divided into disjoint sets of individuals,  $X$ , and values,  $V$ . The interpretation function assigns an element  $v^I \in V$  to each value  $v$ . It assigns a set of binary relations  $a^I$  over  $X \times V$  to each symbol  $a$  in *Attr*, and a set of binary relations  $r^I$  over  $X \times X$  to each symbol  $r$  in *Role*. The extension of a FDL description is defined as follows:*

1. *The extension of a sensor is defined as  $a(v)^I = \{x \in X \mid (x, v^I) \in a^I\}$ . The extension of an existential sensor  $a^I$  is defined as  $\{x \in X \mid \exists v^I \in V \text{ s.t. } (x, v^I) \in a^I\}$ .*
2. *The extension of a role is defined as  $(r D)^I = \{x \in X \mid \exists y(x, y) \in r^I \rightarrow y \in D^I\}$ .*
3. *The extension of a conjunctive expression (AND  $D_1 D_2$ ) is defined as  $D_1^I \cap D_2^I$ .*

We can now define the *subsumption* of a FDL description  $D_1$  by another description  $D_2$ . We say that  $D_1$  *subsumes*  $D_2$  iff the extension of  $D_2$  is a subset of the extension of  $D_1$ , i.e.  $D_1^I \supseteq D_2^I$  for all interpretations  $I$ . To show that FDL allows efficient subsumption, we use the notion of a concept graph that we define next.

## 2.2 Concept Graphs

The notion of concept graphs stems from work in the semantic network and frame-based representation conceived of in the late 1970's and early 1980's. In many ways description logics were invented to provide a concrete semantics for the construction of such graph-based knowledge representations. In our framework they provide a tool for computing subsumption between descriptions and as a convenient representation for learning examples in our learning framework.

FDL concept graphs are a variation on the type invented for [1] to explain "basic CLASSIC". A FDL concept graph is a rooted labeled directed graph  $G = G(N, E, v_0, l_N(*))$ , where  $N$  is a set of nodes,  $v_0 \in N$  is the root of the graph,  $E \subseteq (N \times N \times \text{Role})$  a set of labeled edges (with role symbols as labels) and  $l_N(*)$  is a function that maps each node in  $N$  to a set of sensor descriptions associated with it.

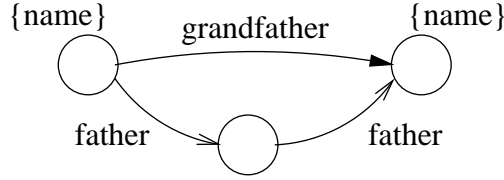
The semantics of FDL concept graphs is defined similarly to that of basic CLASSIC, minus those associated with equality constraints. The extension of a node in the graph is intended to be the set of individuals that can be described by its corresponding description.

**Definition 3.** *Given a FDL concept graph  $G = (N, E, v_0, l_N(*))$ , a node  $n \in N$ , and an interpretation  $I$  in some domain  $\Delta$  composed of elements  $X$  and values  $V$ , we say that an individual  $x \in X$  is in the extension of  $n$  iff:*

1. For each sensor  $a_i(v) \in l_N(n)$ ,  $a_i^I(x, v^I)$  is true. For each sensor  $a_i \in l_N(n)$ ,  $\exists v^I \in V$  s.t.  $a_i^I(x, v^I)$  is true.
2. For each edge  $(n, m, r_i) \in E$ , if  $r_i^I(x, y)$  is true and  $\exists y \in X$  such that  $y$  is in the extension of  $m$ .

As in [6, 1], an individual  $x$  is in the extension of  $G$ , iff it is in the extension of  $n_0$ . It will be clear later that in our paradigm we care about extension only as a clean way to define subsumption; the more basic notion here is the description itself.

*Example 3.* Fig. 2.2 shows the concept graph that corresponds to the descriptions in Ex. 1.



**Fig. 1.** The concept graph for the description in Ex 1.

Two constructs over domain  $\Delta$  are semantically equivalent if they have the same extensions given an interpretation  $I$ . The significance of concept graphs for our purposes stems from the following theorem.

**Theorem 1.** Any FDL description  $D$  is semantically equivalent to an acyclic FDL concept-graph of size polynomial in  $\|D\|$  that can be constructed in polynomial time.

*Proof.* We can recursively construct a concept graph for any FDL description in the following way, similar to the algorithm of [6, 1]:

If the description is some sensor  $a_i$  or  $a_i(v_j)$ , construct a node labeled with the set  $\{a_i\}$  or  $\{a_i(v_j)\}$  respectively.

If the description is a role of the form  $(r_i D)$ , first recursively construct the concept graph for  $D$ . Then add an empty node  $n$  connected to the root of  $D$ 's graph by an edge labeled  $r_i$ . Finally make  $n$  the root of the modified graph.

If the description is of the form  $(\text{AND } D_1 \dots D_n)$ , recursively construct the concept graphs for  $D_1, \dots, D_n$ . Then take the union of the node and edge sets for the above graphs, and merge the roots of all graphs. To merge two nodes  $n_1, n_2$ , construct a new node  $n_{12}$  and add the union of the labels of  $n_1$  and  $n_2$  to it. Then redirect all the edges entering or leaving  $n_1$  and  $n_2$  to  $n_{12}$ . After all roots have been merged, this new node becomes the root of the combined graph.

The result of this polynomial time procedure is a semantically equivalent acyclic concept graph of size polynomial in the number of clauses in the original description, as each clause constructs at most a single node and edge.

Thm 1 allows now to show that FDL supports efficient subsumption queries between descriptions, and moreover that it supports checking subsumption of an arbitrary concept graph by a description.

**Theorem 2.** *For FDL descriptions  $D_1, D_2$  the subsumption of  $D_2$  by  $D_1$  ( $D_1 \supseteq D_2$ ) can be decided in polynomial time. Additionally for a description  $D_1$  and an arbitrary FDL concept graph  $G_2$ , the subsumption of  $G_2$  by  $D_1$  can be decided in polynomial time.*

*Proof.* In order to compute subsumption of  $D_2$  by  $D_1$ , first convert  $D_2$  to a concept graph  $G_2$ . Then, from the root clause of  $D_1$  recursively check each clause against  $G_2$  in the following manner: Set the current clause  $c$  to the root clause in  $D_1$  and the current node  $n$  to  $n_0$  in  $G_2$ . If  $c$  is a sensor of the form  $a_i(v_i)$ , test if  $a_i(v_i)$  is in  $l_N(n)$ . If  $c$  is a sensor of the form  $a_i$ , test if  $a_i$  is in  $l_N(n)$ , or if some  $a_i(v)$  is in  $l_N(n)$  for any value  $v$ .

If  $c$  is a role of the form  $(r_i D)$ , test if an edge labeled  $r_i$  from  $n$  to some  $n'$  exists in  $G_2$ . If so set  $n$  to  $n'$  and  $c$  to the clause  $D$  and recurse. If the check returns false for  $D$  subsuming node  $n'$ , but other edges labeled  $r_i$  exist from  $n$ , repeat the check for each node pointed to from  $n$ . Return true if any check returns true, else return false.

If  $c$  is a clause of the form  $(\text{AND } D_1 \dots D_n)$ , set  $c$  to each clause  $D_i$  and recursively test it on  $n$ . If all tests return true, then return true, else fail.

If the test for the root clause of  $D_1$  returns true, then  $D_1$  subsumes  $D_2$ . This test is obviously linear in the size of  $D_2$  or in the size of a given  $G_2$ . If  $G_2$  is given at the beginning as an arbitrary concept graph, the proof still holds. Note that while  $G_2$  may then be cyclic, the  $D_1$  description by definition is acyclic, so the procedure will terminate.

In our framework subsumption is used to transform a domain element, represented as a concept graph, into a feature set that can serve as an input to a propositional learning algorithm. The efficiency of this operation is crucial to support our efficient learning of relational representations. This result should be contrasted with general hardness results for subsumption [13, 27].

Given these definitions for FDL descriptions and their corresponding concept graph representations, it now becomes possible to describe a feature extraction framework in which such representations play a major role. Efficient subsumption testing will allow the generation of expressive propositional features from arbitrarily complex data represented by concept graphs.

### 2.3 Feature Generating Functions

Up until this point, our treatment of our FDL has closely mirrored that of similar CLASSIC-like DL's [16, 6, 1]. However, our usage of FDL descriptions is vastly different from the usage of descriptions in these other DL's. The most closely related usage may be that of P-CLASSIC descriptions, in which a probabilistic distribution over descriptions is used to perform probabilistic subsumption queries. Instead, in our paradigm, descriptions are used to generate propositional formulae, in a data-driven way via subsumption queries. General propositional learning algorithms can then be

used to learn definitions (classifiers) for predicates in terms of these or to learn distributions over them. We first describe the process of generating propositional formulae using FDL descriptions.

The essential construction of our method is that of a *Feature Generating Function* (FGF), closely related to the RGF of [7].

Our FGF construction, however, differs in an important respect. Here we discuss a general function, whose operation is constrained by the formal syntax of the generating descriptions themselves, having well defined structure and meaning. The FGF notion therefore extends and unifies the related constructions that have used a “relational calculus” to procedurally compose different types of RGFs in order to produce complex features. In fact, we claim that any operation of such a calculus may be pushed onto the syntax of an FDL, and therefore it is possible to define descriptions in our language that produce exactly the same features as produced there.

**Definition 4.** Let  $I$  be some interpretation with domain  $\Delta = (X, V)$ , and let  $\mathcal{I}$  be the space of all interpretations. For a description  $D$  we define a feature  $F_D$  to be a function  $F_D : \mathcal{I} \rightarrow \{0, 1\}$ .  $F_D$  acts as an indicator function over  $\mathcal{I}$ , denoting the interpretations for which the extension of the description  $D$  is not empty.

*Example 4.* A feature can be constructed from any FDL description. E.g., given the sensor description  $D = \text{name}(\text{Thelma})$ ,  $F_D$  might take an interpretation  $I$  in which for  $x \in I$ ,  $\text{name}(x, \text{Thelma}) = \text{true}$ . Thus  $F_D(I) = \text{true}$ .

Given an interpretation  $I$  we say that a feature  $F$  is *active* in  $I$  if it evaluates to true. Such features could be a useful input to a propositional learning algorithm, as they describe hopefully expressive *relational* qualities of the given instance. Generating such features efficiently however is the topic of much debate, as such feature spaces could be prohibitively large or in some cases infinite, making manual generation impossible.

Our next step is to automate the construction of features of this sort. Luckily, the semantics of FDL descriptions and their equivalence to concept graphs gives rise to an efficient method of constructing *active* features, via the notion of the *feature generating function* (FGF).

Let some interpretation  $I$  be represented as a concept graph  $G$ , in which each element of  $I$  is in the extension of some node of  $G$ . To facilitate this construction, we define the inverse of the interpretation function associated with  $I$ . This inverse function is defined to output the value symbol  $v$  associated with each  $v'$ , and to output the attribute or role symbol associated with each binary relation over elements and values. We process the interpretation by creating a node in  $G$  for each element seen in some relation, and processing each relation accordingly. If the relation corresponds to a role symbol, we create an edge between the nodes corresponding to the elements of the relation and label it with that role symbol. If the relation is of the form  $a(x, v)$ , with  $a$  corresponding to some attribute symbol  $a'$ , we add a sensor description  $a'(v')$  to the set  $l(x')$  with the node  $x'$  corresponding to the element  $x$  in the relation. In this way, we create an FDL concept graph  $G$  with each element of  $I$  in the extension of a node in the graph.

Our FGF method takes  $G$  along with a set of FDL descriptions  $\mathcal{D}$ , and outputs a set of active features over  $G$ . The basic method computes the *most-specific-subsumer* (mss)

of  $G$  with respect to each description  $D \in \mathcal{D}$ , and constructs a feature for each mss. The intuition is that each input description defines a “type” of feature, subsuming many possible (partially) ground descriptions over several different interpretations. We say a description is *ground* if it is a description containing only sensors of the form  $a_i(v_i)$ .

**Definition 5.** Let  $\mathcal{F}$  denote an enumerable set of features over the space  $\mathcal{I}$  of interpretations and let  $D$  be a description. A feature generating function  $\mathcal{X}$  is a mapping  $\mathcal{X} : \mathcal{I} \times \mathcal{D} \rightarrow 2^{\mathcal{F}}$  that maps an interpretation  $I$  to a set of all features in  $\mathcal{F}$  such that  $F(I) = 1$  as constrained by the description  $D$ .

Thus, the image of  $I$  under  $\mathcal{X}$  is a re-representation of  $I$  in terms of the (subsumers of the) description  $D$ .

**Definition 6.** The most-specific subsumer of a description  $D$  (or equivalent concept graph  $G$ ) with respect to some generating description  $D_{gen}$  is defined to be the unique ground description  $D_{mss}$  subsuming  $D$  and subsumed by  $D_{gen}$ , containing only ground forms of the sensors in  $D_{gen}$ . In the case that  $D_{gen}$  is itself ground, computing the mss amounts to checking the subsumption of  $D$  by  $D_{gen}$ .

Before discussing the mechanics of construction and evaluating the most-specific subsumer, we give examples of features produced by the mss-based-FGF.

*Example 5.* Given a generating description  $D = (\text{AND name (father (AND age (father name)))})$ , the FGF  $\mathcal{X}_{mss}(D, I)$  would produce the following active features for an extension of the family relations interpretation  $I$  defined in an earlier example.  $F_1 : (\text{AND name(Charles) (father (AND age(52) (father name(Michael)))})$ . This feature represents the fact that in the current interpretation there is an individual whose name is Charles, who is the father of an individual with age 52 who, in turn is the father of some individual with the name Michael.

As usual, the importance of features stems from the fact that they might provide some abstraction. That is, they describe some significant property of the input which may occur also in other, different, inputs.

**Theorem 3.** There exists an mss-based feature generating function  $\mathcal{X}_{mss}$  that, given any interpretation  $I$  represented as a concept graph and a generating description  $D$ , will generate all active features over  $I$  with respect to  $D$  in polynomial time.

*Proof.* We provide an operational definition for  $\mathcal{X}_{mss}$  in terms of its operation on some interpretation  $I$  with respect to  $D$ . Given an  $I$  represented as a concept graph, we first designate one of the nodes  $n$  of  $I$  as the root, and construct the mss of  $n$  as follows:

Test whether  $D$  subsumes  $n$  using the procedure defined above, copying each clause of  $D$  to a separate description  $D_{mss}$  in which each sensor is written as the version ground by the value from the corresponding node of  $I$ . If several values exist for the same sensor at the same node of  $I$ , construct a separate  $D_{mss}$  for each. If we find that  $D$  does not subsume  $n$ , we throw  $D_{mss}$  away. Then for each  $D_{mss}$  write an active feature  $F_{D_{mss}}$ . Repeat this procedure for each node  $n$  of  $I$ .

Since this procedure repeats the linear time subsumption procedure for FDL descriptions for each node in  $I$ , and in the worst case  $I$  and  $D$  may both have  $n$  nodes/clauses, the entire algorithm takes  $O(n^2)$  time for each interpretation.

The importance of providing a formal framework for the feature generating process, via the FGF constructions stems from the fact that this notion generalizes essentially all ad-hoc ways in which people have been defining complex features in applications. For example, all “ngram”-like features that are used in NL applications with a variety of algorithms [21] can be easily defined via an FDL, and produced via the feature generating function construction. The current framework allows for the definition of significantly more expressive features, with a clear semantic and syntax, and with efficient inference procedure. As an example, there is a large body of literature in NLP on feature languages developed as an alternative, rich representation of natural language [3]. This very interesting but somewhat stagnant direction could be pushed forward if, for example, these languages are phrased as FDLs, and incorporated into the framework presented here via the notion of FGFs. This will allow learning and representing complex and probabilistic predicates in terms of descriptions in the languages.

### 3 Experiments

As mentioned earlier, the problem of learning a definition for kinship relations has been an arch-typical example of relational learning since the late 80’s/early 90’s [12, 23]. Simply put, given two members of a family, the task is to decide what their relation is.

The data set used is the same set of 112 positive relations over two separate families originally used in [12], and later described in [23]. These relations are of the form *mother(Penelope,Arthur)* etc. and describe the kinship relationship of the first person to the second. In previous ILP experiments regarding this problem, a rule set is incrementally generated to cover more of the predicates seen in the data. In our experiments we take a markedly different approach, as we must formulate the data as examples for an on-line learning algorithm.

With a formal notion of concept graphs and descriptions in place, we can describe the setup of our learning experiments. We map all 112 relations of the aforementioned family relations data to a single interpretation  $I$  represented as a concept graph, where we define a node for each person in the data set and a directed edge for each relation between two people.

In general, our learning system will process each interpretation, generating features as described earlier. Each feature is denoted by a description, serving as a unique lexical index. Each vector of features, called an *observation*, is passed to the learning algorithm, which treats one (or more) of them as a class label and the rest as features to be learned from. As features are generated in a data-driven way from a possibly infinite attribute space [2], observations will necessarily be variable-length vectors of features. Using the SNoW system allows us to learn effectively from this form of input.

To learn a definition for each target relation, we first generate an observation for each relation between two nodes in the data instance. This brings up the important notion of *focus of attention*. In many learning scenarios such as the family relations scenario, representing the data in a graph-based manner allows us to set a focus of attention based on the target predicate to be learned. This focus, for our purposes, takes the form of a node or a pair of nodes. The focus of attention allows us to restrict the range of nodes from which to produce features for the current observation based on

certain graph properties in the interpretation - *e.g.* proximity to the focus. Thus, we can represent all of our family data as a single graph, and based on which nodes are targeted, can extract different meaningful features for each observation. The method we use to restrict features produced based on locality is described below.

In theory, our locality restriction is similar to the restriction made in [10] of features based on “context-dependent node attributes of depth  $n$ ” and “context-dependent edge attributes of depth  $n$ ”. We restrict the range of features produced by restricting the set of nodes that can act as roots of features. This is accomplished by adding an operator to our standard FDL syntax as follows:

Given a description  $D$  and interpretation  $I$ , we say that the expression  $(\mathbf{IN}(n) \mathbf{D})$  is a description denoting the set of individuals within  $n$  role edges from the focus of attention of  $I$ . We then add an argument to our FGF function  $\mathcal{X}$  to denote the focus of attention for  $I$ , and write it as  $\mathcal{X}(I, D, (i, j))$ , where  $D$  contains an  $\mathbf{IN}$  clause and  $i$  and  $j$  are specifications of nodes in  $I$ .  $\mathcal{X}$  is then evaluated in the following manner: If  $i = j$  and  $n$  is positive, only generate features for nodes up to  $n$  edges away from  $i$  along some path leading from  $i$ . If  $n$  is negative then only generate features for nodes up to  $n$  edges away from  $i$  along some path leading to  $i$ . If  $i \neq j$ , then follow the same procedure, except for negative  $n$  test if it is along the path to  $i$ , and for positive  $n$  test if it is along the path from  $j$ . If  $n = 0$ , then only generate features for  $i$  along any path to node  $j$ .

Now we describe the 3 sets of features generated for each observation. We extracted a feature for each single edge between the the focus nodes, corresponding to the class label, via the locality restriction measure defined above. We then extracted sets of features corresponding to each labeled path of size 2 and 3 between the focus nodes to serve as features. Our first set of generating descriptions are of the form  $(\mathbf{IN}(0) (rel \ \&))$ , where *rel* is each of the 12 family relations in the data. Features produced from this description will serve as class labels in the observations. The second set is of the form  $(\mathbf{IN}(0) (rel (rel \ \&)))$  and the third of the form  $(\mathbf{IN}(0) (rel (rel (rel \ \&))))$ , where *rel* is again each of the 12 relations. These descriptions will produce the features from which the classifier will be learned.

We trained and tested the SNoW classifier with our relational features in a winner-take-all manner. Thus for each class label we obtain a linear function  $\sum_i w_i \cdot F_i(I)$  of weighted features, with indices corresponding to lexical FDL descriptions. For example,  $F_i$  might be  $(\mathbf{IN}(0) (brother (mother \ \&)))$ , or  $(\mathbf{IN}(0) (husband (sister (father \ \&))))$ , which could be associated with the *uncle* class label.

We train and test only on pairs connected by valid relations, which correspond to the 112 positive examples. This method is similar to the approaches taken in [12, 23, 24] where only positive examples along with a handpicked set of negative examples were used in training and testing.

## 4 Results & Discussion

For our experiment, we compare our performance with the FOIL system of [23] and the relational pathfinding enhanced FORTE system presented in [24]. Training and testing sets were selected randomly using an incremental partitioning of the 112 positive ex-

amples. Accuracy was computed for each partitioning by averaging over 20 runs over a random selection of training and test examples. For the first experiment, after two cycles of training on 101 examples, our accuracy on the majority of runs was %100, with an error on a few runs. These few errors were due to the test set not covering some relation because of the randomization, and on average over all the runs our accuracy converged to %99.36.

By comparison, the FOIL system achieved an accuracy of 97.5% after performing 500 sweeps of 100 training examples and testing on the remainder averaging over 20 runs. The pathfinding version of the FORTE system also achieved 100% accuracy, but after training over a set of 300 randomly chosen positive and negative examples.

Our experiment showcases the ability of our system to match the output level of existing ILP systems, while exhibiting a drastic increase in efficiency both in learning and evaluation. Several factors contribute to this increase as well as to the high level of accuracy maintained.

First, as we have shown, efficient constructions for generating and evaluating formulae make feature construction easy. Instead of conducting a search for good bindings during the learning process, all active formulae are generated up-front as propositions, and feature efficient propositional algorithms take care of learning good definitions for the functions. Second, the expressivity of our learned functions for these targets is actually *greater* than that of competing ILP methods. Due to the restrictions necessary for ILP methods to be tractable, rules learned in systems such as FOIL are defined over  $k$ -DNF, for a small  $k$ . FDL descriptions allow the generation of clauses similar to those making up a  $k$ -DNF, and our learned functions are linear functions over these clauses.

Supporters of the ILP paradigm also purport that a disjunctive set of rules is easier to interpret after learning than a network of features and weights. Yet in this case as all features are clauses similar to those induced by an ILP algorithm, it is a simple matter to examine the final learned weight vector to see what clauses contribute heavily to the learned function. For example, for the target feature **(IN(0) (uncle &))** in our final weight vector the feature **(IN(0) (husband (aunt &)))** achieves a high weight, while the feature **(IN(0) (brother (mother &)))** has a slightly lower weight but still contributes to the final activation in cases where the person corresponding to the intermediate node is not married.

It is possible to imagine a number of easy extensions to the basic problem of learning family relations that would easily integrate into our existing experimental framework. For example, one might wish to learn a binary classifier to predict the gender of any individual from the family data, purely based on the relations between individuals. Such an attribute could be represented with a sensor description on any node in the family data, and using features similar to those extracted for the previous experiment, a classifier could just as easily be learned.

In addition, instead of training our family relation classifier in a winner-take-all fashion, we could just as easily train individual binary classifiers for each relation. Thus we would treat each observation extracted for a pairing of nodes as a positive example for the classifier for that relation, and negative for all others. In this manner, we could also train a binary classifier for a non-relation target that could predict the lack of a relation between any two individuals, and use it as a filtering stage, only attempting to

further classify those pairings of nodes not predicted to be non-related. Such a filtering stage is similar to the construction of [25], and would be needed in any “real-world” implementation of the family relation prediction problem in which we would presumably receive an arbitrary pairing of two individuals and need to determine how they were related.

## 5 Conclusion

We presented a paradigm for efficient learning and inference with relational data. This paradigm defined the notion of feature description logics - a relational language with clear syntax and semantics that can be used, via feature generation functions, to efficiently re-represent world observations in a way that is suitable for general purpose learning algorithms. We have shown that this allows one to efficiently learn complex relational representations, in a system in which the basic components are articulated in a language whose structure and meaning are well understood.

While we have concentrated on outlining the approach for a specific FDL, it is important to point out that a wide family of FDLs can be used within our framework. In fact, other CLASSIC-like description logics as well as their probabilistic variations, can be incorporated into the framework with the addition of a Feature Generating Function, and participate as building blocks in the process of learning relations and predicates. For example, features generated by FGFs need not be defined as Boolean. They can be associated with a real number, indicating the probability the feature holds in the interpretation, and can be used by the learning algorithm, allowing an immediate use of P-classic like languages. This approach provides a different view on ways to extend such description languages - orthogonal to the one suggested by existing extensions, such as PRMs [9], that are more suitable to relational database-like (probabilistic) inferences and do not provide a natural solution to learning predicates and relational structure as in the examples shown here.

We have also shown the suitability of our framework for learning in purely relational domains by examining it in the context of the classical ILP problem of learning family relationships. In this simple example, we show the viability and flexibility of our propositionalization approach by exhibiting superior performance results both in terms of accuracy and speed. A variation of our framework has also been applied successfully to problems in the realm of Information Extraction.

This work extends and unifies several lines of works in KRR, learning and natural language processing, and provides ways to build on existing work in these areas and use it in ways that provide hope for a coherent usage of learning and inference methods for large scale intelligent inference.

## References

1. P. F. Patel-Schneider A. Borgida. A semantics and complete algorithm for subsumption in the classic description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.
2. A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, 1992.

3. Bob Carpenter. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.
4. W. Cohen. PAC-learning recursive logic programs: Efficient algorithms. *Journal of Artificial Intelligence Research*, 2:501–539, 1995.
5. W. Cohen. PAC-learning recursive logic programs: Negative result. *Journal of Artificial Intelligence Research*, 2:541–573, 1995.
6. W. W. Cohen and H. Hirsh. Learnability of description logics with equality constraints. *Machine Learning*, 17(2/3):169–200, 1994.
7. C. Cumby and D. Roth. Relational representations that facilitate learning. In *Proc. of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 425–434, 2000.
8. S. Dzeroski, S. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proceedings of the Conference on Computational Learning Theory*, pages 128–135, Pittsburgh, PA, 1992. ACM Press.
9. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, 1999.
10. P. Geibel and F. Wyszotzki. Relational learning with decision trees. In *European Conference on Artificial Intelligence*, pages 428–432, 1996.
11. A. R. Golding and D. Roth. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130, 1999. Special Issue on Machine Learning and Natural Language.
12. G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12, Amherst, Mass, August 1986.
13. D. Kapur and P. Narendran. NP-completeness of the set unification and matching problems. In *Proc. of the 8th conference on Automated Ddeduction*, volume 230, pages 489–495. Springer Verlag, 1986.
14. K. Kersting and L. De Raedt. Bayesian logic programs. In J. Cussens and A. Frisch, editors, *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pages 138–155, 2000.
15. R. Khardon, D. Roth, and L. G. Valiant. Relational learning for NLP using linear threshold elements. In *Proc. of the International Joint Conference of Artificial Intelligence*, 1999.
16. D. Koller, A. Levy, and A. Pfeffer. P-classic: A tractable probabilistic description logic. In *Proc. of the National Conference on Artificial Intelligence*, pages 360–397, 1997.
17. S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*. Springer Verlag, 2001.
18. H. Levesque and R. Brachman. A fundamental tradeoff in knowledge representation and reasoning. In R. Brachman and H. Levesque, editors, *Readings in Knowledge Representation*. Morgan Kaufman, 1985.
19. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
20. J. W. Lloyd. *Foundations of Logic Programming*. Springer-verlag, 1987.
21. L. Mangu and E. Brill. Automatic rule acquisition for spelling correction. In *Proc. of the International Conference on Machine Learning*, pages 734–741, 1997.
22. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 20:629–679, 1994.
23. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
24. B. L. Richards and R. J. Mooney. Learning relations by pathfinding. In *National Conference on Artificial Intelligence*, pages 50–55, 1992.

25. D. Roth and W. Yih. Relational learning via propositional algorithms: An information extraction case study. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1257–1263, 2001. Acceptance Rate: 197/796 (25%).
26. M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proc. of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 421–431, Boston (USA), 1989.
27. M. Sebag and C. Rouveirol. Any-time relational reasoning: Resource-bounded induction and deduction through stochastic matching. *Machine Learning*. To Appear.
28. B. Selman. *Tractable Default Reasoning*. PhD thesis, Department of Computer Science, University of Toronto, 1990.
29. L. G. Valiant. Robust logic. In *Proceedings of the Annual ACM Symp. on the Theory of Computing*, 1999.