A Logical Framework for Graph Theoretical Decision Tree Learning

Peter Geibel and Fritz Wysotzki

Methods of Artificial Intelligence, Computer Science Department, Sekr. Fr 5–8
Technical University Berlin, Franklinstr. 28/29, D-10587 Berlin, Germany
Email {geibel|wysotzki}@cs.tu-berlin.de

Abstract. We present a logical approach to graph theoretical learning that is based on using alphabetic substitutions for modelling graph morphisms. A classified graph is represented by a definite clause that possesses variables of the sort node for representing nodes and atoms for representing the edges. In contrast to the standard logical semantics, different node variables are assumed to denote different objects. The use of an alphabetical subsumption relation (α-subsumption) implies that the least generalization of clauses (α-generalization) has different properties than Plotkin's least generalization (lgg). We present a method for constructing optimal α-generalizations from Plotkin's least generalization. The developed framework is used in the relational decision tree algorithm TRITOP.

1 Introduction

In this paper, we present a logical approach to graph theoretical decision tree learning that is based on using alphabetic substitutions for modelling graph morphisms. The nodes of a graph are represented by variables of the sort (type) node while atoms are used for representing the edges. A classified graph is this way represented by a definite clause. In contrast to the standard logical semantics, different node variables are assumed to denote different objects which is quite natural in many, e.g. technical application domains.

To compare the generality of clauses in this framework, we will introduce an alphabetical subsumption relation (α-subsumption) that relies on alphabetic substitutions (α-substitutions) and preserves the distinctness of node variables. In [5], N. Helft uses inequalities to express the distinctness of variables. Though we also use inequalities for defining the meaning of clauses containing node variables, we propose a different notion of least generalization. In contrast to Plotkin's (and Helft's) least generalization (lgg, see [10]), each such least α-generalization is reduced, has less literals than the original clauses, but is not unique. Nevertheless, optimal α-generalizations with a high structural complexity can be constructed from Plotkin's least generalization. We will show that the α-generalization algorithm can also be used for testing α-subsumption, as well as for computing the number of possible embeddings of one structure into another, which is crucial for defining relational decision trees.
In this framework, domain theories can be included by a modified version of saturation ([13]). This approach is similarly used in the system CILGG ([7]) and can be shown to be equivalent to a modified version Buntine's generalized subsumption ([11]).

For showing the feasibility of our approach, it is used in the learning system TRITOP that induces relational decision trees that contain complex structural (relational) attributes as tests. TRITOP is based on the decision tree algorithms CAL3 ([16]) and ID3 ([12]). In TRITOP, the relational attributes are constructed from the examples by combining the computation of \( \alpha \)-generalizations with the application of refinement operators. In contrast to the graph based decision tree learner INDIGO ([4]), the attributes are not computed before, but during the construction of the tree which leads to smaller and more accurate decision trees.

This article is organized as follows. In section 2, the representation of classified relational structures and of attributes by definite clauses is discussed, and \( \alpha \)-subsumption is introduced. In section 3 we describe an approach to using domain theories. In section 4 an algorithm for computing optimal \( \alpha \)-generalizations is given. In section 5 the learning system TRITOP is sketched. Section 6 concludes.

2 Training Examples and Attributes

For learning a \( n \)-ary relational concept, a training set \( \mathcal{E} = \{ E_i \mid 1 \leq i \leq e \} \) is given, that contains \( e \geq 0 \) definite clauses \( E_i = \text{class}(x_1, \ldots, x_n, c_i) \leftarrow a_1^i, \ldots, a_m^i \) from a function free and sorted logical language ([9]). In each example \( E_i \), the body \( a_1^i, \ldots, a_m^i \) is a conjunction of atoms \( a_j^i \) containing the classified objects \( x_1, \ldots, x_n \) and contextual objects as variables of the sort \( \text{node} \). The class value \( c_i \) of the tuple \( x_1, \ldots, x_n \) stems from the set \( \{ c_1, \ldots, c_v \} \) of class constants that posses the sort \( \text{Class} \). The concept to be learned is characterized by the predicate symbol \( \text{class} \) with type \( \text{node} \times \ldots \times \text{node} \times \text{Class} \) whose arity \( n + 1 \) determines the arity \( n \) of the concept.

In the following, we will use the relations \( b(x) \) ("\( x \) is a block"), \( s(x, y) \) ("\( x \) supports \( y \)"), and \( d(x, y) \) ("\( x \) does not touch \( y \)"), for describing blocks world situations. Suppose that we want to learn the binary relation \( \text{passage} \) that is true for two objects that both support the roof of a blocks world arch. We will consider the training set \( \mathcal{E} = \{ E_1, E_2, E_3, E_4 \} \) that contains the following examples and counterexamples for the concept \( \text{passage} \): \( E_1 = \text{class}(x_1, x_2, 1) \leftarrow b(x_1), b(x_2), b(x_3), s(x_1, x_2), s(x_2, x_3), d(x_1, x_2), E_2 = \text{class}(x_1, x_2, 1) \leftarrow b(x_1), b(x_2), b(x_3), b(x_4), s(x_1, x_3), s(x_2, x_3), d(x_1, x_2), d(x_2, x_4), E_3 = \text{class}(x_1, x_2, 0) \leftarrow b(x_1), b(x_2), b(x_3), s(x_1, x_3), s(x_2, x_3), E_4 = \text{class}(x_1, x_2, 0) \leftarrow b(x_1), b(x_2), b(x_3), b(x_4), s(x_1, x_3), s(x_2, x_3), d(x_1, x_2), d(x_4, x_1), s(x_4, x_2).

To construct a relational decision tree, we can use relational attributes that are implicitly given by the set of all substructures of the examples in the training set \( \mathcal{E} \). For example, the attribute \( A = \text{class}(y_1, y_2, y_3) \leftarrow s(y_1, y_3), s(y_2, y_3), d(y_1, y_2) \) describes the objects \( y_1 \) and \( y_2 \) by their relation to a third contextual object \( y_3 \). \( A \) occurs as a substructure in the examples \( E_1, E_2, \) and \( E_4 \) but not