Changing Labels in the Double-Pushout Approach Can Be Treated Categorically

Hans J. Schneider

Universität Erlangen-Nürnberg – Institut für Informatik (Lehrstuhl 2),
Erlangen, Germany
schneider@informatik.uni-erlangen.de

Abstract. In the double-pushout approach to graph transformations, most authors assume the left-hand side to be injective, since the noninjective case leads to ambiguous results. Taking into consideration productions that change labels, however, may add ambiguity even in the case of injective graph productions. A well-known solution to this problem is restricting the categorical treatment to the underlying graphs, whereas the labels on the derived graph are defined by other means. In this paper, we resume the detailed results on arbitrary left-hand sides that Ehrig and Kreowski have already given in 1976. We apply these results to the case of relabeling such that we can retain the elegant categorical constructions at the level of labeled graphs.

1 Introduction

Graph structures are ubiquitous in computer science as well as in many application areas. They are a very natural way to explain complex situations on an intuitive level, and they are used to define the syntax of structures according to given rules (graph grammars) as well as to describe dynamically deriving new situations from given ones (graph transformations). In 1973, we have introduced the double-pushout approach to formalize this process [9]. The approach allows replacing substructures in an intuitive way immediately instead of encoding the graphs into strings and then applying string transformations. Using concepts from category theory leads to elegant proofs, e.g., concerning local Church-Rosser property, parallelism, and concurrency, which can easily be applied to different categories [3]. Whereas the pushout construction is straightforward in the categories of interest, we need a pushout complement on the left-hand side of the double-pushout that may be ambiguous or not even exist.

The construction is well-understood in $\mathcal{S}et$ and $\mathcal{G}raph$. We have already outlined in our first paper that the category $\mathcal{L}graph$ of labeled graphs with label preserving morphisms does not add any new difficulties to constructing pushouts or pushout complements. Many applications of interest, however, use morphisms that change not only the structure of the graph, but also change some labels. Examples can be found, e.g., in data bases [11, 16], compiler technique [15], term graph rewriting [12], asynchronous processes [17–19], and so on. To some extent, this is possible even in $\mathcal{L}graph$: The node to be relabeled is not included in the

H.-J. Kreowski et al. (Eds.): Formal Methods (Ehrig Festschrift), LNCS 3393, pp. 134–149, 2005.

interface graph, and therefore, corresponding nodes on the left-hand and on the right-hand side can bear different labels. But, this node must have a fixed context given in the production, and it is not applicable in other contexts. In [9], we have overcome the limitations of label-preserving graph morphisms, by not labeling the interface graph and the corresponding nodes and edges in the context graph. The pushout construction is performed in $\mathcal{G}raph$, and a so-called "labeled gluing" ensures that the derivation includes only labeled graphs. *Rosen* has shown that this approach can be written in $\mathcal{L}graph$ using two different morphisms from the interface graph into the context graph [13]. With respect to term graph rewriting, *Habel* and *Plump* solve relabeling by considering partially labeled graphs [10]: If a node of the interface graph is not labeled, the corresponding nodes on the left-hand side and on the right-hand side may be labeled differently. Furthermore, their construction allows the left-hand side and the right-hand side to be partially labeled. In this case, the production represents a (possibly infinite) set of productions, which you get by labeling the unlabeled nodes with any element of a given set of symbols. Another approach that deserves special attention has been introduced by *Parisi-Presicce*, *Ehrig*, and *Montanari* [11]. With data base applications in mind, these authors define a structure on the labeling alphabet that allows the user to specify which changes are possible and which are not. Morphisms in the category $\mathcal{SL}graph$ are graph morphisms compatible with this structure. Both approaches define the labels on the derived graph by set-theoretic means.

In this paper, we show that the $\mathcal{SL}graph$-approach is able to model various applications that need relabeling or specifying sets of productions in a uniform way. Especially, it is possible to retain the elegant categorical constructions at the level of labeled graphs. Unfortunately, the structured alphabet adds a new difficulty in constructing the pushout complement: The solution may be unambiguous even in the case of injective graph morphisms. But this problem can be treated on a categorical level using the results by *Ehrig* and *Kreowski* concerning $\mathcal{E}-\mathcal{M}$-factorizable categories [6].

The rest of the paper is organized as follows. In the next section, we summarize the basic definitions of derivability in general and in the category of structurally labeled graphs. We recapitulate the decomposition theorem that allows us to characterize the pushout complements in the category of structurally labeled graphs (Section 3). Finally, Section 4 considers two special cases that are important in many applications, namely (infinite) sets of productions and productions changing some labels. The results, however, are formulated independent of these applications.

2 Background

In the double-pushout approach (see, e.g., [3]), the notion of derivability with respect to arbitrary categories is defined a symmetrical diagram: