(Co)algebraic specification and its application in XML-based modelling

Jens Lechner

Lehrstuhl Informatik 1 - Logik in der Informatik
Technische Universität Dortmund
Otto-Hahn-Str. 12, 44227 Dortmund

jens.lechner@cs.uni-dortmund.de
https://ls1-www.cs.uni-dortmund.de/de/

Abstract. XML documents specified by a DTD are modelled as terms of the constructive signature corresponding to the DTD. The terms are (possibly infinite) edge- and node-labelled trees regarded as partial functions with a countable domain and a finite codomain. This representation allows us to define the semantics of XPath expressions in a natural way and thus gives the opportunity to solve certain complexity questions in the context of XML and XPath using effective descriptive set theory.

Keywords: DTD, XML, XPath, signature, term, edge- and node-labelled tree, complexity, effective descriptive set theory

A document type definition (DTD) gives rise to a constructive signature $\Sigma = (S, \mathcal{I}, C)$ in the sense of [5–7]. Here S is a set of sorts, $\mathcal{I}$ a set of sets of indices and C a set of constructors $c : e \to s$ where e a type built up from S, $\mathcal{I}$ and $\mathcal{I}$-indexed sums and products. Documents satisfying the DTD are modelled as elements of the set CT_Σ of finite or infinite ground Σ-terms. In particular, *infinite* terms model the unfoldings of documents with hyperlinks. Hence it is only *rational* terms (unique solutions of iterative equations [1]) that are needed in this application area.

As shown in [5–7], each constructive signature Σ can be turned into a destructive signature $co\Sigma$ such that CT_Σ is a final $co\Sigma$-algebra. In general, the carriers of final models of destructive signatures can the represented as sets of ground *coterms*, i.e., partial functions t from words over destructors and (product) indices to ϵ and (sum) indices, t can be thought of as a (finite or infinite) typed tree. A node n of t is labelled with ϵ if it is the root of an S-typed subtree. Otherwise n is labelled with a sum index. Edges are labelled with destructors or product indices.

The logical core of XPath [2, 3] is described by a context-free grammar whose abstract syntax is – as any CFG – a constructive signature, say Π. Ground Π-terms model XPath expressions. The semantics of XPath is the Π-algebra Sem whose carrier is, rougly speaking, given by the function space

$$CT_\Sigma \to (X^* \to \mathcal{P}(X^*))$$

where X is the set of edge labels of the trees that represent elements of CT_Σ. Since each node of $t \in CT_\Sigma$ is uniquely represented by word over X, the folding of an XPath expression $\alpha \in T_\Pi$ in *Sem* maps t to the relation between the nodes of t that is determined by α.

Given an XPath expression $\alpha \in T_\Pi$ and a document $t \in CT_\Sigma$, one can ask for the complexity of $fold^{Sem}(\alpha)(t)$ relative to the complexity of t. This is a question I want to answer and I guess that the answer is that $fold^{Sem}(\alpha)(t)$ is co-recursive enumerable relative to t.

Other questions that arise are the containment problem and the satisfiability problem. The containment problem asks whether for two XPath expressions α and β, all $t \in CT_\Sigma$ and all $w \in X^*$ $fold^{Sem}(\alpha)(t)(w)$ is a subset of $fold^{Sem}(\beta)(t)(w)$. The satisfiability problem asks whether for an XPath expression α, $t \in CT_\Sigma$ and $w \in X^*$ $fold^{Sem}(\alpha)(t)(w)$ is empty. For these questions I also want to classify their complexity. I think that effective descriptive set theory [4] can be helpful in answering these two questions.

References

1. J. Adamek, S. Milius, J. Velebil, *Free iterative theories: a coalgebraic view*, Math. Struct. in Comp. Science (2003) 259320
2. G. Gottlob, C. Koch, and R. Pichler, *XPath Processing in a Nutshell*, SIGMOD Record 32, No. 2 (2003) 21-27
3. M. Marx, M. de Rijke, *Semantic Characterizations of Navigational XPath*, SIGMOD Record 34, No. 2 (2005) 41-46
4. Y. N. Moschovakis, *Descriptive Set Theory*, North Holland 1980
5. P. Padawitz, *From grammars and automata to algebras and coalgebras*, Proc. CAI 2011, Springer LNCS 6742 (2011) 21-43
6. P. Padawitz, *Übersetzerbau (Algebraic compiler construction)*, course notes, TU Dortmund 2016
7. P. Padawitz, *Fixpoints, Categories, and (Co)Algebraic Modeling*, course notes, TU Dortmund 2016