

# Using monads to fuse recursive programs

Claus Jürgensen<sup>1</sup>

Claus.Juergensen@Inf.TU-Dresden.DE

Faculty of Computer Science  
Dresden University of Technology  
D-01062 Dresden, Germany

— extended abstract —

We try to combine the ‘syntactic composition of tree transducers’ [FV98, KV01] on the one hand side and ‘short cut fusion’ [GLP93] on the other hand side.

Short cut fusion is based on the ‘cata/build-rule’ [Joh01] or ‘acid rain theorem’ [TM95]. Therefore it is necessary to represent the recursive functions as catamorphisms. A catamorphism is a generalization of the well known list-function `foldr` for arbitrary algebraic datatypes. In terms of category theory a catamorphism is the unique mediating morphism from an initial algebra.

We invented a new fusion technique using monads: instead of a catamorphism we use the unique mediating morphism from a free monad.

Consider the small Haskell program:

```
data Nat    = Zero | Succ Nat
data Bool   = True  | False

even Zero   = True
even(Succ n) = odd n
odd Zero    = False
odd(Succ n) = even n
```

The latter four equations define the two mutually recursive functions `even` and `odd`. We can view this system of equations as a function which maps the left-hand-side-expressions onto the according right-hand-side-expressions:

$$\varrho_X : \mathbf{Q}(\Sigma X) \rightarrow \mathbf{T}_\Delta(\mathbf{Q}X)$$

where  $X = \{\mathbf{n}\}$  is the set of variables. The endofunctors  $\Sigma$ ,  $\Delta$ , and  $\mathbf{Q}$  describe the application of ranked symbols from  $\{\mathbf{Zero}^{(0)}, \mathbf{Succ}^{(1)}\}$ ,  $\{\mathbf{True}^{(0)}, \mathbf{False}^{(0)}\}$ , and  $\{\mathbf{even}^{(1)}, \mathbf{odd}^{(1)}\}$ , respectively, to a set (e.g.  $\Sigma X = \{\mathbf{Zero}\} \cup \{\mathbf{Succ } x \mid x \in X\}$ ). The functor  $\mathbf{T}_\Delta$  constructs all  $\Delta$ -trees over a set:  $\mathbf{T}_\Delta X \cong X + \Delta(\mathbf{T}_\Delta X)$ .

It is easy to show that the function  $\varrho_X$  is natural in  $X$  and thus we have a natural transformation:

$$\varrho : \mathbf{Q} \cdot \Sigma \rightrightarrows \mathbf{T}_\Delta \cdot \mathbf{Q}$$

which we call the **rule** of the functional program. It is now possible (using some category theory magic like adjoint functors) to equivalently transform this rule into the form:

$$\varrho' : \Sigma \rightrightarrows |\mathbf{HT}_\Delta|$$

---

<sup>1</sup> Supported by the postgraduate program ‘Specification of discrete processes and systems of processes by operational models and logics’ (GRK 334/2) of the German Research Community (DFG)

where  $\mathbb{T}_\Delta = (\mathbb{T}_\Delta, \eta, \mu)$  denotes the free monad over  $\Delta$ ,  $\mathbb{H}$  is a monad transformer, and  $|\cdot|$  the forgetful functor mapping a monad onto its underlying endofunctor. A rule in the latter form is the main ingredient of a so called **monad transducer**.

Using the universal properties of free monads we can define a semantics for monad transducers. Moreover, we have proved a new fusion theorem for monad transducers similar to the ‘acid rain theorem’ or ‘cata/build-rule’.

All the above has been motivated by the theory of tree transducers: A tree transducer [Rou68] is a finite tree automaton with in- and output. Its integral part is a set of rules. Some classes of tree transducers can be viewed as syntactic fragments of functional programming languages. Our example Haskell program is a top-down tree transducer which has the two states **even** and **odd**.

In the theory of tree transducers there exist many composition results, which are all of the form: ‘If two tree transducers belong to the syntactic classes  $X$  and  $Y$ , respectively, then the composition of the two belongs to class  $Z$ .’

The composition of top-down tree transducers is an instance of short cut fusion [JV01]. But for more complicated tree transducers we have not been able to apply short cut fusion, so we invented the monad transducer.

Using our monad transducer we can describe many kinds of tree transducers in a uniform way. The expressiveness of monads makes it possible to conveniently treat tree transducers which have context parameters. We have reproduced many known composition results (*e.g.* for high-level tree transducers [EV88]), but with much easier proofs. In the future we will describe so called tree-series transducers [EFV02] as monad transducers which will lead us to new composition results.

Furthermore, we want to implement our new fusion technique by means of the rewriting rules of the GHC [PTH01] so that it can be used to deforest functional programs. Whether this can (easily) be done and/or whether this will increase program efficiency are open questions.

## References

- [EFV02] J. Engelfriet, Z. Fülöp, and H. Vogler. Bottom-up and top-down tree series transformations. *J. Automata, Languages and Combinatorics*, 2002. accepted.
- [EV88] J. Engelfriet and H. Vogler. High level tree transducers and iterated push-down tree transducers. *Acta Informatica*, 26:131–192, 1988.
- [FV98] Z. Fülöp and H. Vogler. *Syntax-directed semantics—Formal models based on tree transducers*. Monographs in Theoretical Computer Science, An EATCS Series. Springer-Verlag, 1998.
- [GLP93] A. Gill, J. Launchbury, and S. L. Peyton-Jones. A short cut to deforestation. In *Proceedings of Functional Programming Languages and Computer Architecture (FPCA '93)*, pages 223–232, Copenhagen, Denmark, June 1993. ACM Press.
- [Joh01] P. Johann. Short cut fusion: Proved and improved. In W. Taha, editor, *Proceedings of the 2nd International Workshop on Semantics, Applications, and Implementation of Program Generation (SAIG 2001)*, volume 2196 of *LNCS*, pages 47–71, Florence, Italy, September 2001. Springer.
- [JV01] C. Jürgensen and H. Vogler. Syntactic composition of top-down tree transducers is short cut fusion. Technical Report TUD-FI01-10, Technische Universität Dresden, Fakultät Informatik, D-01062 Dresden, Germany, November 2001.

- [KV01] A. Kühnemann and J. Voigtländer. Tree transducer composition as deforestation method for functional programs. Technical Report TUD-FI01-07, Technische Universität Dresden, Fakultät Informatik, D-01062 Dresden, Germany, August 2001.
- [PTH01] S. L. Peyton-Jones, A. Tolmach, and T. Hoare. Playing by the rules: Rewriting as a practical optimisation technique in GHC. In Ralf Hinze, editor, *Preliminary Proceedings of the 2001 ACM SIGPLAN Haskell Workshop (HW '2001)*, pages 203–233, Firenze, Italy, September 2001.
- [Rou68] W. C. Rounds. *Trees, transducers and transformations*. PhD thesis, Stanford University, 1968.
- [TM95] A. Takano and E. Meijer. Shortcut deforestation in calculational form. In *Proceedings of the Conference on Functional Programming Languages and Computer Architecture*, pages 306–313, La Jolla, CA, June 1995. ACM Press.