Interpretation Orders for Applicative Term Rewriting

2110100 Teppei Tanaka

Term rewriting is a computation model that is based on pattern matching with directed equations, called rewrite rules. In this model, we regard sets of rewrite rules, called term rewrite systems, as declared programs. In particular, term rewrite systems over constant symbols and a unique binary function symbol are called *applicative term rewrite systems*.

Consider the following applicative term rewrite system.

$$\begin{aligned} \mathsf{add}\ 0\ y &\to y & \mathsf{map}\ f\ \mathsf{nil} &\to \mathsf{nil} \\ \mathsf{add}\ (\mathsf{s}\ x)\ y &\to \mathsf{s}\ (\mathsf{add}\ x\ y) & \mathsf{map}\ f\ (:\ x\ \ell) &\to :\ (f\ x)\ (\mathsf{map}\ f\ \ell) \\ \mathsf{double}\ x &\to \mathsf{add}\ x\ x \end{aligned}$$

The term $\mathsf{map}\ \mathsf{double}\ (:\ 0\ (:\ (\mathsf{s}\ 0)\ \mathsf{nil}))$ is rewritten as follows:

$$\begin{aligned} & \mathsf{map}\ \mathsf{double}\ (:\ 0\ (:\ (\mathsf{s}\ 0)\ \mathsf{nil})) \\ \to\ & :\ (\mathsf{double}\ 0)\ (\mathsf{map}\ \mathsf{double}\ (:\ (\mathsf{s}\ 0)\ \mathsf{nil})) \\ \to\ & :\ (\mathsf{add}\ 0\ 0)\ (\mathsf{map}\ \mathsf{double}\ (:\ (\mathsf{s}\ 0)\ \mathsf{nil})) \\ \to\ & :\ 0\ (\mathsf{map}\ \mathsf{double}\ (:\ (\mathsf{s}\ 0)\ \mathsf{nil})) \\ \to\ & :\ 0\ (:\ (\mathsf{double}\ (\mathsf{s}\ 0))\ (\mathsf{map}\ \mathsf{double}\ \mathsf{nil})) \\ \to\ & :\ 0\ (:\ (\mathsf{add}\ (\mathsf{s}\ 0)\ (\mathsf{s}\ 0))\ (\mathsf{map}\ \mathsf{double}\ \mathsf{nil})) \\ \to\ & :\ 0\ (:\ (\mathsf{s}\ (\mathsf{add}\ 0\ (\mathsf{s}\ 0)))\ (\mathsf{map}\ \mathsf{double}\ \mathsf{nil})) \\ \to\ & :\ 0\ (:\ (\mathsf{s}\ (\mathsf{s}\ 0))\ (\mathsf{map}\ \mathsf{double}\ \mathsf{nil})) \\ \to\ & :\ 0\ (:\ (\mathsf{s}\ (\mathsf{s}\ 0))\ \mathsf{nil}) \end{aligned}$$

The last term cannot be rewritten any more. Such a term is called normal form, that is regarded as a computational result. Applicative Term rewrite systems underlie functional programming languages and automated theorem provers that handle higher-order logic.

Termination is the property that any term can reach normal forms eventually. In order to show termination, *reduction orders* are employed. If all rules orient from left to right for some reduction order then its system is terminating. For example, suppose that the above ATRS is oriented by some reduction order $>$ as follows:

$$\begin{aligned} \mathsf{add}\ 0\ y &> y & \mathsf{map}\ f\ \mathsf{nil} &> \mathsf{nil} \\ \mathsf{add}\ (\mathsf{s}\ x)\ y &> \mathsf{s}\ (\mathsf{add}\ x\ y) & \mathsf{map}\ f\ (:\ x\ \ell) &> :\ (f\ x)\ (\mathsf{map}\ f\ \ell) \\ \mathsf{double}\ x &> \mathsf{add}\ x\ x \end{aligned}$$

Then we have the system is terminating. Reduction orders are the basis of not only termination proofs, but also automated theorem provers.

However, it is hard for existing reduction orders to orient the direction that variables are duplicated, therefore $\mathsf{map}\ f\ (:\ x\ \ell) > :\ (f\ x)\ (\mathsf{map}\ f\ \ell)$ and $\mathsf{double}\ x > \mathsf{add}\ x\ x$ are difficult. The aim of this research is to find a new reduction order that can handle variable duplication.

In this thesis we introduce the reduction order by combining interpretation order with *uncurrying* (Hirokawa et al. 2013). Uncurrying is a powerful transformation method for termination proof. Here we illustrate our method by using the above example. Our method allows us to uncurry and interpret at the same time. For termination of the original systems, we just have to find an interpretation order $>$ that orient as follows:

$$\begin{aligned} \mathsf{add}_2(0, y) &> y & \mathsf{map}_2(f, \mathsf{nil}) &> \mathsf{nil} \\ \mathsf{add}_2(\mathsf{s}_1(x), y) &> \mathsf{s}_1(\mathsf{add}_2(x, y)) & \mathsf{map}_2(f, :_2(x, \ell)) &> :_2(f\ x, \mathsf{map}_2(f, \ell)) \\ \mathsf{double}_1(x) &> \mathsf{add}_2(x, x) \end{aligned}$$

Actually, a few conditions are added for the algebra, but it is easier to find interpretations than the original one.

In addition to the definition of new interpretation order, we found an automatable termination criterion by using *polynomial interpretation* (Lankford, 1979). Here we list the contributions of the thesis:

- a new reduction order and
- its automation technique.

Experiments on the Termination Problem Database (TPDB) show effectiveness of our new reduction order.