On Proving Termination of Higher-Order Rewrite Systems by Dependency Pair Technique

Masahiko Sakai[1] and Keiichirou Kusakari[2]

[1] Department of Information Engineering, Nagoya University,
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
sakai@nuie.nagoya-u.ac.jp
[2] Research Institute of Electrical Communication, Tohoku University,
Katahira, Aoba-ku, Sendai, 980-8577, Japan
kusakari@nue.riec.tohoku.ac.jp

We will discuss the dependency pair (DP) technique for proving termination of Nipkow's higher-order rewrite systems (HRS). The DP technique is effective on term rewriting systems, because it gives us a mechanical support for proving non-simple termination. Sakai, Watanabe and Sakabe tried to extend the DP method to the higher-order case[1]. Since it requires a reduction quasi-ordering having the subterm property, the argument filtering method is not applicable any more that weakens their approach extremely. After that, we proposed another DP method[2] on HRS by introducing a notion of dependency forest and show that the termination property of a higher-order rewrite system R can be checked by the non-existence of an infinite R-chain, if R is non-duplicating or non-nested. Although no longer the subterm property is needed hence the argument filtering is applicable, the strong restriction, non-duplicating or non-nested, is required. We explain this method and discuss the possibility to weaken the restriction.

The key point of the DP technique is producing R' by adding rules, called DPs, to the given TRS R in order to transform infinite reduction sequence of R to an infinite reduction sequence of R' having infinite head-reductions, called DP-chain. Thus, proving termination is reduced to showing that no infinite DP-chain exists. In the transformation of the sequence, we take an appropriate subterm of each term in the original sequence. Generally, the head symbols in DP-chain are introduced by right-hand sides in R. Fortunately, it is easy to define DPs in first-order case, because each additional rules are obtained by replacing the right-hand side of a original rule with its subterms headed by defined symbol. In higher-order case, it is difficult to define DPs having such a good property, because the head symbols in DP-chain may be carried or even copied as instances of higher-order variables after their introduction. We introduce the notion of dependency forest in order to analyze and to trace dependencies between the introduction and the consumption of such symbols.

The following example shows why the restriction non-duplication or non-nested is required. Consider an HRS R;

$$R \;=\; \begin{cases} i(X) \to g(X, X), \\ g(h(\lambda x.F(x)), X) \to F(X) \end{cases}$$

Although we have only one dependency pair $\langle i^{\#}(X), g^{\#}(X, X)\rangle$, the following infinite reduction sequence exists; $i(h(\lambda x.i(x))) \to g(h(\lambda x.i(x)), h(\lambda x.i(x))) \to i(h(\lambda x.i(x))) \to \cdots$.

[1] Masahiko Sakai, Yoshitsugu Watanabe, Toshiki Sakabe, An Extension of Dependency Pair Method for Proving Termination of Higher-Order Rewrite Systems, IEICE Trans. on Information and Systems, Vol.E84-D, No.8, pp.1025–1032, 2001.
[2] Masahiko Sakai, Keiichirou Kusakari, On New Dependency Pair Method for Proving Termination of Higher-Order Rewrite Systems, The International Workshop on Rewriting in Proof and Computation (RPC'01), Sendai, Japan, October 25–27, pp.176–187, 2001.