International Journal of Innovative
Computing, Information and Control
Volume **5**, Number **9**, September **2009**

 ISSN 1349-4198
pp. 2463–2479

GENERATING FUNCTIONALITY-BASED RULES FOR PROGRAM CONSTRUCTION

Katsunori Miura[1], Kiyoshi Akama[1] and Hiroshi Mabuchi[2]

[1]Information Initiative Center
Hokkaido University
Sapporo, Hokkaido, 060-0811, Japan
kmiura@uva.cims.hokudai.ac.jp; akama@iic.hokudai.ac.jp

[2]Faculty of Software and Information Science
Iwate Prefectural University
Takizawa, Iwate, 020-0193, Japan
mabu@soft.iwate-pu.ac.jp

Received July 2008; revised November 2008

Abstract. *In the Equivalent Transformation (ET) computation model, a program is constructed by the successive accumulation of ET rules. A method of correct ET rule generation by meta-computation has already been proposed. However, although the method covers a broad range in the generation of ET rules, all the important ET rules are not necessarily generated. Generation of more ET rules can be achieved by supplementing generation methods which are specialized for important classes. A class of functionality-based rules is one of those classes. A functionality-based rule describes a procedure in which two different variables included in two atoms are equalized based on the functionality of the atoms. In this paper, we propose an algorithm that systematically and recursively generates functionality-based rules and discuss its effectiveness in the synthesis of ET programs. A functionality-based rule is generated based on proving a logical formula consisting of the given atoms and dis-equality. The proof is computed by utilizing several ET rules and the rule which will be obtained by the algorithm. The proposed algorithm together with meta-computation can synthesize programs to solve difficult problems including those that cannot be solved by definite clause Logic programs.*
Keywords: Equivalent transformation, Rule generation algorithm, Functionality-based, ET rule, Inductively-used rule

1. **Introduction.** As software systems become larger and more complex, the need for the development of cost-effective and reliable software increases. As a result, attention is increasingly being paid to component-wise program construction [4, 5, 16]. This method constructs a reliable program by accumulating program components, each of which has passable independence, one by one. In the Equivalent Transformation (ET) computation model [1], a program is a set of ET rules; an ET rule effectively achieves component-wise program construction [12].

[**The importance of ET rules in program construction**] An *ET rule* is a type of rewriting rule which describes a procedure to replace one clause set with another clause set having an equivalent declarative meaning. Since each ET rule is completely *independent* and *individually* correct, a program can be constructed simply by accumulating ET rules one by one. Additionally, since the correctness of an ET rule with respect to a specification can be proven, it is guaranteed that a set of ET rules is a correct program with respect to a specification. An ET rule is a precise program component, and is the minimum unit for describing a procedure. Since an ET rule has high expressive power for describing a