JAIST Repository
https://dspace.jaist.ac.jp/

Title	型に基づくパターンマッチングコンパイル方式の構築と実装
Author(s)	纓坂, 智
Citation	
Issue Date	2004-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1794
Rights	
Description	Supervisor:大堀 淳, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Type-based formulation of pattern matching compilation and its implementation

Satoshi Osaka (210016)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 13, 2004

Keywords: pattern matching, compilation, type, decision tree.

Pattern matching is a mechanism of functional languages, including ML, Haskell, and OCaml. It allows the programmer to distinguish and to use the values of complex types simply by writing the desired set of patterns. In order to achieve the branches of the structured values, a compiler translates this high-level primitive into an sequence of primitive operations consisting of equality test for atomic types, tag test for union types, and sequential decomposition for product types. This process is one of the most elaborate parts of a compiler of a functional language; and various ways of it have been proposed. One of the major approaches is to construct a "decision-tree".

Compared to the other approaches, This approach has better features such as to generate efficient code that performs the same test only once for any given input, and to detect redundancy and non-exhaustiveness of a given set of patterns. However, a decision tree model which literature has been treated indicates the strategies only for a set of essential patterns, and is not sufficient to construct a pattern matching compiler for a practical languages. It is not at all clear how to construct a decision tree for a set of practical patterns, how to detect redundancy and exhaustiveness, or how to construct binding. The correctness of the algorithm also has not been shown. Our result can shed some lights these problems.

Copyright © 2004 by Satoshi Osaka

The goal of this paper is to establish a type theoretic basis for the pattern matching compilation, and to impliment a practical pattern matching compiler. In order to achieve this goal, we go forward with our research by following strategies.

1. We define a denotational semantics for pattern matching. We view a given set of patterns as a disjoint set of subsets of the terms that partition the domain of a given type, and regard a pattern matching construct as a primitive to determine the subset to which a given term belongs. This view allows us to define a denotational semantics for pattern matching. We can derive a pattern matching algorithm and can prove its correntness and other properties.

2. We define tree representations for the denotational semantics and derive a pattern matching compilation algorithm. The algorithm we derived is a basic principle for pattern matching compilation. We improve the algorithm so as to compile a pattern matching expression efficiently and to allow easy implementation of a pattern matching compiler for a practical language.

3. We prove the correctness of the algorithm and other properties. We prove that a tree which has been constructed by the algorithm is correct with respect to the denotational semantics we have defined. We can then prove that the algorithm is correct and can detect redundancy and non-exhaustiveness of a given patterns.

Our second goal is to implement a practical pattern matching compiler based on above type theoretical basis. The compiler has been implemented for the full set of Standard ML pattern languages and has been extended for or-pattern. Additionally, We have introduced some optimization techniqus such as to make the compilation faster and to generate efficient code. The pattern matching compiler we have implemented will be a part of an ML compiler which we have been developing in JAIST.