論 文 の 内 容 の 要 旨

論文題目

A Platform for Composable and Statically-Checkable Domain-Specific Languages

（合成可能かつ静的検査可能なドメイン専用言語のためのプラットフォーム）

氏　　名　　市川　和央

This dissertation proposes ProteaJ, which is a programming language supporting composable and flexible user-defined operators and user-defined scope rules for them. User-defined operators, named protean operators, are procedures with their own syntax. Protean operators are overloaded by operators that have the same syntax but have different return type or operand types. An operator is available only at an expression where its return type is expected. In other words, programmers can regard return type and operand types as non-terminal symbols. Programmers can define various syntax as user-defined operators by exploiting types as non-terminals. This helps programmers implementing embedded domain specific languages (embedded DSLs). Users can safely use multiple such the embedded DSLs that different programmers developed even if the embedded DSLs have operators that have similar syntax. Programmers do not have to care about conflicts of syntax since protean operators are distinguished by types. We call this property composability. Composability is important for software development style that uses multiple DSLs together.

ProteaJ also supports context-sensitive expressions, which are a variant of lambda expressions, for expressing name binding and scope rules. Context-sensitive expressions take parameters but the parameters are not explicitly written. Since the parameter names are not given, the parameters cannot be accessed via the parameter names. Instead, the members of a parameter are available without receivers in context-sensitive expressions. In ProteaJ, protean operators can be instance members. This means that we can express syntax that is available only at context-sensitive expressions. Hence, we can express name

binding and scope rules by using context-sensitive expressions. ProteaJ supports turnstile types, which express the types of context-sensitive expressions. ProteaJ also supports generic names to recognize arbitrary names given by end-users. User-defined language constructs implemented by using context-sensitive expressions can be safely composed because the scope of protean operators is expressed by turnstile types.

Additionally, ProteaJ allows programmers to define declarations with their own scope rules. A declaration declares something and enables to use it after the declaration. ProteaJ provides an `activate` statement that takes an object as its argument and enables its members after the statement without receivers. The scope of activated members does not follow the scope of local variables in the host language. If the `activate` statement is used in the body of an operator/method, the visible members are also available at the call-site of the operator/method. Hence, we can implement an operator that simulates a variable declaration. The scope of activated members is given by `scope for` clauses. A `scope for` clause takes several types and it expresses the scope of activated members of given objects. Programmers can implement different scope rules for each embedded DSLs. To statically check which members are activated, ProteaJ also provides `activates` clauses. Since the compiler can check what members are available by checking signatures of methods/operators, user-defined declarations and their scope rules can be safely composed.