Abstraction for asynchronous message passing on Java

Osamu Abe

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

February 14, 1997

Keywords: programming language, object-oriented concurrent model, future, Java, distributed computation.

Purpose and background

In this research, we extend Java to the language that can treat concurrent computations more abstractly and execute our language on the Java virtual machine.

Computer network gains in popularity rapidly and the range of the application of concurrent computations has increased rapidly now, too. However, most of these developments still rely on the paradigm of sequential computations. Especially in computer languages. For instance, most of the language that is used to develop the application actually combines mechanisms of process, thread or coroutine with the language that is developed for sequential computations.

However, the paradigm of sequential computations has several problems when it applied to concurrent computations. For instance, there are problems of the synchronization, the data sharing, and the exclusive control of critical sections. It is difficult to treat these problems in the paradigm. If you write the program with a concurrent computation using the language designed for the paradigm of sequential computations, the program is not intuitively understandable for man.

Java is an object-oriented language that designed for distributed environment used with Web browser. Java is a superior language for this reason. However Java is not a language suitable for concurrent computations either. Though, the system that treats distributed objects abstractly on Java named RMI and HORB has been developed. But there is no system that can treat concurrent computations abstractly. The mechanism of treating concurrent computations easily is important in the distributed computation.

Object-oriented concurrent language MILK

We designed and implemented a language called MILK, which is an extension of Java. MILK combines an object-oriented concurrent model with Java to treat concurrent computations abstractly. Each object can be operated independently and concurrently in the object-oriented concurrent model. Therefore, it is possible to consider by personifying the object. It is easy for man to think about the concurrency intuitively, and programs can be easily designed by this model.

In the object-oriented concurrent model, computations are done by passing the asynchronous message between objects. We adopt two kinds of ways of asynchronous message passing in MILK. These ways are called: a past type and a future type. The past type is that the sender continues the computation without waiting for the process termination in the receiver when a message passing is done between objects. After a sending message, the value is not returned from the receiver. As the case of the future type, the place holder that is called a future is passed from the sender to the receiver with the message and the sender continues the computation at once as well as the past type. The receiver delivers the result of the process to the future. In the case that the sender needs the result, the future is examined. If the result returns, the sender uses the value. If not, the sender waits until the value returns.

MILK can be converted into Java by the translator we made in this research. First, the translator divides the source code into a sequence of tokens by the scanner. The sequence of tokens is passed to the parser and the syntax tree is made by the parser. Each node of the tree is constructed from objects that maintain necessary information on each syntax. Then, the translator traces this syntax tree. When the translator detects the syntax that is extended from Java, the syntax is translated to the code that can be operate by Java. Finally, the translator makes a character string from the syntax tree, and outputs the strings to a file.

The definition of the future and basics of concurrent objects are prepared beforehand as class libraries. Syntax that is different from the syntax of Java is replaced with the codes that can be operated by Java. The class libraries that are prepared beforehand are used in this codes. All concurrent objects are derived from the class defined in the class library. Concurrent objects have a message queue and a thread as components. These components are defined in the class defined in the library. The asynchronous message passing is done by using these components. The class definition of the future is designed that the synchronization is correctly done.

In the distributed environment, these class libraries are additionally made beforehand for each distributed system that is used with MILK. The translation is done by selecting the class library for the distributed system. Any distributed object that the distributed system treats can be used at once as a concurrent object by doing so.

Evaluation and prospect

The concurrent object and the asynchronous message passing of the past type and the future type were put into Java by MILK. As a result, one can intuitively express concurrent

computations. MILK succeeds to the merits that Java has because MILK is the extension of Java. So Milk is superior to Java in programming easily.

However, there are many limitations according to the limit of the translator. The translator solves the only reference of the name that is defined in the same class definition. For instance, the reference to the future defined in other class files is not able to be solved, so it is prohibited. Moreover, because of the incompleteness of the object of Java the information of the concurrent object is not encapsulated perfectly. These problems cannot be solved by the method translating the target code into the Java source code.

However, such a problem can be technically solved by making the compiler that generates the Java class file directly. The way achieved by the compiler is not restricted by the syntax of Java. Therefore the concurrent computation can be expressed more flexible way.

The VM of Java is not yet implemented on a parallel machine. Therefore, the concurrent object cannot demonstrate the real value in the meaning of the truth. However, when thinking about the speed of advancement and the popularization of the hardware in recent years, it can be thought that the day when parallel machine gains in popularity is not so far. If becoming so, the scene that the object-oriented concurrent language like MILK acts might also increase.