KLICとJavaのメッセージインタフェース
Message interface between KLIC and Java

飯塚 和久† 田中 二郎††
Kazuhisa IIZUKA Jiro TANAKA

†筑波大学修士課程理工学研究科
Master's Program in Science and Engineering, University of Tsukuba
††筑波大学電子情報工学系
Information Science and Electronics, University of Tsukuba

概 要

KLIC と Java はそれぞれ特色を持ったプログラミング言語であり、２つの言語を利用することによりアプリケーションの幅が広まると期待される。我々は、これらの言語間で通信を簡単に行うためのメッセージインタフェースの作成を行った。このインタフェースを利用することにより、KLIC プログラムの GUI として Java アプレットを用いたアプリケーションの記述等が容易に行うことができる。本稿では、今回実装したデータストリームや、手続き呼び出し型の通信、Java のオブジェクト操作について説明し、その実装についてふれる。

1 はじめに

1.1 背景と目的

最近のインターネットの急速な普及により、プログラミング言語 Java[1]に大きな注目が集まっている。しかし、Java は GUI やインターネット関係のプログラミングには適していると言われているが、他のプログラミングの分野に、必ずしも向いているとは限らない。エキスパートシステムなどに代表される知識処理の分野では、論理型言語と呼ばれるクラスの言語が適しているといわれており、KLIC[2]もこのクラスの言語の１つである。

ここで、知識処理分野のアプリケーションで、知識処理の計算を KLIC で、GUI 部を Java でといったような、２つのプログラミング言語にまたがったものを考えてみる。このようなシステムの場合、２つの言語間で、なんらかの通信を行う必要があり、これらを行う適当なインタフェースが必要になる。

本稿では、このような KLIC と Java の２言語間の通信をメッセージパッシングとして扱い、そのメッセージをやり取りする各言語上のインタフェースに関して考察を行う。

1.2 KLIC について

ここで、KLIC について簡単に説明を行う。KLIC は、正確に言えば、並列論理型言語と呼ばれる言語の１つである GHC(Guarded Horn Clauses) [3]の実行環境であるが、今回は、”KLIC”という言葉をプログラミング言語と同一に扱う。

KLIC は、ガード付きホーン節と呼ばれるもので構成され、ゴールのリダクションと、ユニフィケーションにより計算、実行が行われる。実際のプログラミングにおいては、「プロセス」と「ストリーム」という２つの概念を用いてプログラムが行われる。

ここで、プロセスとは、並列に動作するプログラムの中で、その計算がある程度の連続性を持った、まとまりのある処理がある場合に、それら個々の処

理の流れを指す。また、ストリームとは、これらのプロセス間の通信路を意味している。プログラムは、このようなプロセスをいくつか生成し、ストリームによって連結して構成されるのである。

2　メッセージインタフェース

2.1　データストリーム

まず、簡単なメッセージ通信として、ストリーム通信によるデータのやり取りを考える。これは、一方の言語側でストリームにデータ書き込むと、もう一方の言語側でこのデータを受け取ることができるようなものである。今回のインタフェースでは、Java と KLIC の間でいくつかのデータ型が使用できるようにすることで、利用しやすいメッセージ通信を提供する。以降、このストリームによる通信形態をデータストリームと呼ぶことにする。

データストリームで提供するデータ型であるが、双方の言語で共通に利用するということを考慮し、扱いやすい KLIC のデータ型を利用する。まず、基本データ型として、アトム、整数、文字列がある。また、ファンクタ構造を構造データ型として提供することにする。なお、Java 側では、これらのデータを扱うためのクラスを用意し、これらのクラスのインスタンスを用いてデータを扱うようにした。

2.2　手続き呼び出し

アプリケーション間で通信を行う際、その通信手順において、ある特定のプロトコルが多用される場合がある。その代表的なものとして、手続き呼び出し型の通信(図１)[4]がある。KLIC と Java のアプリケーション間の両方向においても、このような形の通信が考えられる。今回は、データストリームであげたデータ型が利用可能な手続き呼び出し型のメッセージ通信を提供する。

KLIC から Java への手続き呼び出しとして、Java のクラスメソッドについて考えてみる。クラスメソッドは、オブジェクトを特定しなくても、いつでも利用できる。またメソッド呼び出しは、手続き呼び出しの形をしているので、簡単に利用することが可能となる。

一方、KLIC では、1.2節で述べたように、プロセスとストリームを用いてプログラミングを行うため、一般的な手続き呼び出しの形式は存在しない。

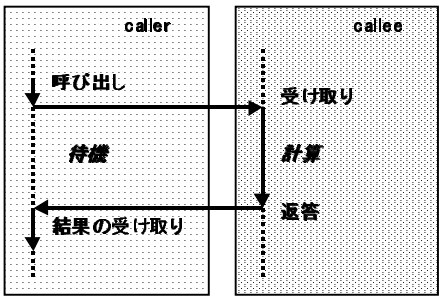

図１　手続き呼び出しの流れ

、一般的な手続き呼び出しの形式は存在しない。そこで、KLIC のプロセスを生成する際に、入力ストリームに引き数となるデータを設定し、リダクション(計算)を行い、その計算結果を出力ストリームに返すようなプロセスを考えてみる。このような特別なふるまいをするプロセスを利用すれば、Java から KLIC への手続き呼び出し型のメッセージ通信として利用することが可能となる。

2.3　KLIC から Java オブジェクトへの操作

Java のプログラムに対して詳細な操作を行うためには、個々のオブジェクトに対して操作を行う必要がある。そこで、KLIC 側から Java の個々のオブジェクトへの操作、つまり、オブジェクトの作成、作成したオブジェクトに対する、メソッド呼び出し、フィールドへのアクセス等を可能にするインタフェースを提供する。

まず、KLIC と Java は、言語パラダイムが異なるため、KLIC 側で Java のオブジェクトをどのように扱うかについて検討する必要がある。ここで注目するのは、Java のオブジェクトの振る舞いである。オブジェクトは、内部状態を持ち、他との通信を行うようなものである。このような性質は、KLIC のプロセスとよく似ている。そこで、KLIC では、Java のオブジェクトを１つのプロセスとしてみなす[5]ことにより、Java のオブジェクトを、KLIC のプログラミングと適合した扱いやすいインタフェースが実現可能となる。

Java のオブジェクトを作成する際は、KLIC 側にそのオブジェクトに対応するプロセスを作成することに対応する。この時、Java 側では、指定されたクラスのオブジェクトを作成し、オブジェクトを管理

するためのテーブルにこのオブジェクトを保存させる。オブジェクトに対してメソッドを呼び出したり、フィールドにアクセスする際は、KLIC 側の対応するプロセスにメッセージを送ることが対応する。このプロセスは、送られたメッセージを Java 側のオブジェクトテーブルに伝え、指定されたオブジェクトのメソッドの呼び出し等を行えばよい。

3 実装

3.1 全体構成

様々なメッセージ形態を実現するに際し、各アプリケーション間でデータをやり取りする下位レベルの通信路が必要になる。KLIC と Java の各アプリケーションは、それぞれ別々のプログラムとして動作している。また、Java のプログラムはアプレットとして異なったホスト上で実行される場合もある。このようなことから、インタフェースを実現するための下位レベルのメッセージ通信としてソケットを利用することとした。

メッセージインタフェースの全体的な構成は図２のようになる。各アプリケーションは、各言語上に用意されたのインタフェースを通じてメッセージを交換することとなる。この各言語上のインタフェースは定められたプロトコルを用いて、ソケットを用いてデータを送受することによりメッセージインタフェースが実現される。

3.2 各インタフェース

KLIC 側のインタフェースは大きく分けて三つの部分からなる(図３)。”main”は、KLIC プログラムからの様々な指示を受ける部分であり、データストリームへの出力は、この部分でデータを文字列に変換しソケットに送出する。”spool”は、手続き呼び出しなどの返り値を設定する変数を記憶する部分である。”input”は Java 側でソケットに書き込まれた文字列を読み込み、必要な操作を行う。例えば、手続き呼び出しの戻り値が渡された場合は spool にその値を渡し、手続き呼び出しを完了させる。

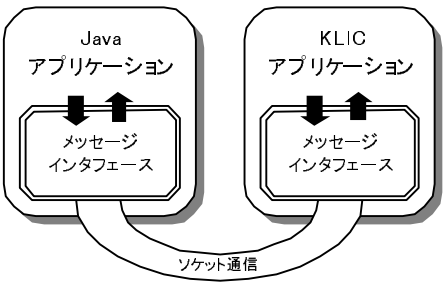

図２ 全体構成

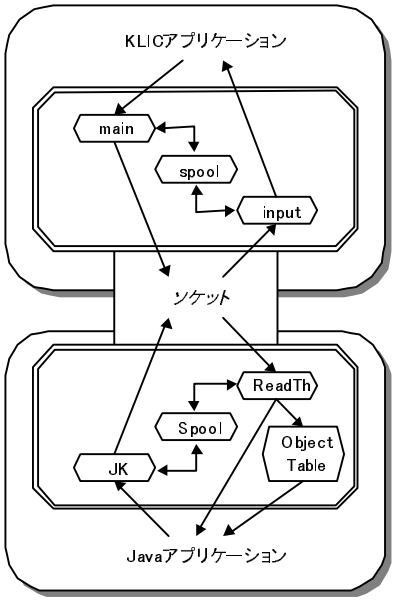

図３ インタフェースの内部構成

Java 側は大きく４つの部分に分かれており、”JK”, “Spool”, “ReadTh” は、それぞれ KLIC 側の ”main”, “spool”, “input” に対応する。また、”Object Table” は、オブジェクトを管理するテーブルである。

4 例題

今回作成したメッセージインタフェースを用いた例として、いくつかのアプリケーションのを作成した。

KLIC のデータ I/O(図４)は、KLIC のプログラムに対するデータの入出力を Java の GUI で行えるようにしたものである。これは、今回作成したデータストリームのインタフェースで接続したものである。

これを利用するには、例えば図５に示したような KLIC プログラムを記述すればよい。このプログラムの場合、Java の GUI から入力された２つのデータを受け取り、その計算結果を GUI で表示する。

ハノイの塔を解くアプリケーション(図６)は、Java から KLIC への手続き呼び出しを利用したものである。KLIC 側では図７に示すようなハノイの

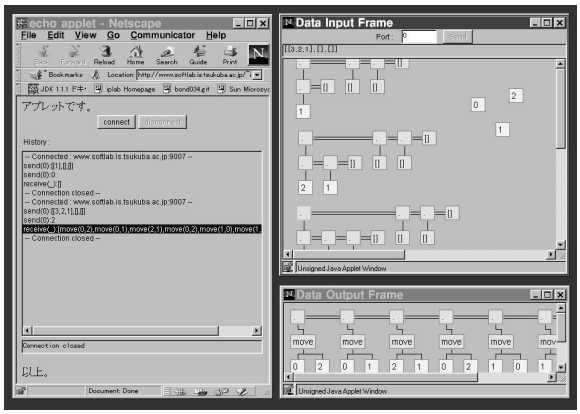

図４ 例（データ I/O）

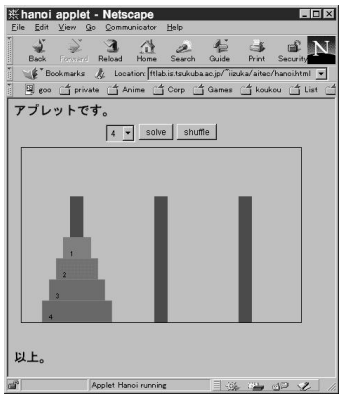

図６ 例（ハノイの塔）

```
main :-
        ......
   jkstream:jkstream(JKS, Sock),
   JKS = [get(Data1), get(Data2)|Tail],
   calc(Data1, Data2, Result),
   main2(Tail, Result).

main2(JKS, Result) :- wait(Result) |
   Tail = [put(Result), close].

calc(Data1, Data2, Result) :-
        ......
```

図５ データストリームを用いたプログラム

```
public void actionPerformed(ActionEvent e) {
  Data args[] = new Data[] {
     Data.Integer(n),  Data.Integer(0),
     Data.Integer(1),  Data.Integer(2)};
  Data ret = jk.call("hanoi", "hanoi", args);
  New Animation(ret).       start();
}
```

図８ Java から KLIC のプログラムの呼び出し

```
:- module hanoi.

hanoi(N, From, Acc, To, Ans) :- N =:= 0 |
   Ans = [].
hanoi(N, From, Acc, To, Ans) :- 1 =< N |
   NN := N - 1,
   hanoi(NN, From, To, Acc, Ans1),
   hanoi(NN, Acc, From, To, Ans2),
   append(Ans1, [move(From, To) |Ans2], Ans).
```

図７ ハノイの塔を解く KLIC プログラム

ある。KLIC 側では図７に示すようなハノイの塔を解くプログラムを記述してあり、これを、Java Applet で記述された GUI 部から利用する。GUI 部から、ハノイの塔を解く手順を取得する際は、図８に示すような手続き呼び出しのインタフェースを利用している。

なお、呼び出される KLIC 側のプロセスになるゴールは、複数の入力引数と、１つの出力引数を持たなければならない。これらは、手続き呼び出しの引き数と戻り値に利用されるストリームとなる。図５のプログラムでは、*N, From, Acc, To* のそれぞれが入力引数に当たり、*Ans* が出力引数に当たる。

5　まとめ

KLIC と Java の間のメッセージ通信について考察し、データストリーム、手続き呼び出し方の通信、そして KLIC から Java のオブジェクトを操作するための手法について説明を行った。また、システムの実装と、このインターフェースを利用した例を示した。今後の課題としては、KLIC において”未完成メッセージ”を用いたストリーム通信が利用されるが、これを Java 側から利用できるインタフェースの実現を行いたいと考えている。

参 考 文 献

[1] Java Technology Home Page. available from http://www.javasoft.com/
[2] KLIC's page. available from http://www.icot.or.jp/AITEC/COLUMN/KLIC/klic-J.html
[3] Kazunori Ueda. Guarded Horn Clauses. ICOT Technical Report TR-103, ICOT, 1985
[4] Andrew D. Birrell, Bruce Jay Nelson, “Implementing Remote Procedure Calls”, ACM Transactions on Computer System, Vol.2, No.1, February 1984, pp.39-59
[5] Shapiro, E. and Takeuchi, A. “Object Oriented Programming in Concurrent Prolog”, New Generation Computing, Vol.1, No.1, 1983, pp.25-48