A-015

動的な接続関係を持つJavaプログラムの一記述法とπ計算への変換

Descripting Java Programs with dynamic link and Translating them into π-processes

山口 将志　加藤 暢　樋口 昌宏
Masashi Yamaguchi　Toru Kato　Masahiro Higuchi

1．はじめに

複数のJavaオブジェクトが並行動作するシステムでは、通信の際オブジェクトの参照渡しを用いることにより、オブジェクトが通信相手を動的に切り替えることができ、柔軟なシステム構成が可能となる。

オブジェクトの参照渡しに対応する機能として、π計算[1]には通信相手の指定に用いられる名前(通信ポート)を受け渡す機能がある。我々はこの点に着目し、Javaのオブジェクト間通信を表現する形式モデルとしてπ計算が適当であると考えた。π計算プロセス式に対する検証系としてMWB[3]が開発されており、Javaプログラムもπ計算プロセス式に変換することによりMWBを用いた形式的な検証が可能となると考えられる。本報告では、Javaプログラムからπ計算プロセス式への具体的な変換手法を提案する。また、実際に通信動作を行う実用的なJavaアプリケーションを作成し、提案する手法を用いてπ計算プロセスに変換した結果について述べる。

2．π計算

π計算は通信ポートを介してプロセスが値をやり取りするプロセス代数の一種である。通信ポートそのものも値としてやり取りすることができる。

$$a(x).P \mid \bar{a}b.Q \quad \xrightarrow{\tau} \quad P\{b/x\} \mid Q$$

上記の式は、同期出力$\bar{a}b$と同期入力$a(x)$が同期して実行され、実行後P中のポート名xがbに置き換えられることを表している。π計算で連接演算子"."、選択(SUM)演算子"+"、並行演算子"|"、条件(MATCH)演算子"if ... then ..."を持ち"+"、"if then"のSemanticsを以下に示す。

$$\text{SUM}\ \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \text{MATCH}\ \frac{P \xrightarrow{\alpha} P'}{if\ x=x\ then\ P \xrightarrow{\alpha} P'}$$

3．Javaからπ計算への変換

Javaプログラムからπ計算プロセス式へ変換するために、Javaオブジェクト間のメッセージの受け渡しをπ計算の同期出力、同期入力で表現する。メッセージの送受信はJavaプログラムのメソッド呼び出し、メソッドの戻り値の返送、他オブジェクトのフィールド参照の時に発生する。そこで、Javaプログラム中のこれら送受信に該当する文を全てπ計算動作式として表現する。具体的にどのように変換するかを、以下の例(図1)で説明する。

```
class A{          class B{
   int ai;              Server_proxy sv = new
   int a(){ ...            Server_proxy(SrvID);
   return m;            A x = sv.broker(ObjA);
   }                    int t = x.a();
}                       x.ai = 1;
                     }
```

$$classA = (\nu\ ap)\ ap(rp,c).(if\ c = ma\ \ then\ M_a \\ +if\ c = fai\ \ then\ \ F_{ai}).classA$$

$$classB = X_a.X_{ai}$$

図 1:　Javaプログラム→π計算プロセス式

Server_proxyは並行に動作しているすべてのインスタンスへの参照を管理するためのクラスである。Aはリソースを提供するクラス、Bはリソースを使用するクラスである。クラスBのインスタンスはServer_proxyクラスのインスタンスに問い合わせることで、クラスAのインスタンスへの参照を得ることができる。変換の詳しい内容を下記に記す。

classA:　*ap*は*classA*固有の参照受け付け用ポートであり、呼び出した側を参照を意味するポート名*rp*と参照されるフィールド名またはメソッド名*c*を受け取る。フィールドai定義は、参照時の出力処理と入力処理が存在するためπ計算プロセスに変換すると

$$F_{ai} = rp(ai) + \overline{rp}\ ai$$

となる。M_aはクラスAのaというメソッドの動作を意味している。メソッドaの最後にはreturn m;があるので出力動作とし、フィールド定義と同様に受信用ポートを考慮にれて、

$$M_a = P.\overline{rp}\ m$$

近畿大学大学院総合理工学研究科
エレクトロニクス系工学専攻

となる。P はメソッド a の内部動作を表している。以上のように、$classA$ は ap や rp を用いてオブジェクトの識別が可能なため希望するオブジェクトのメソッドやフィールドを利用することができる。

$classB$: class B の最初の行は、すべてのオブジェクトの参照を管理しているサーバへの通信準備である。A x = sv.broker(ObjA); で、このサーバに対し HORB[2] の機能を使って classA のインスタンスの参照を問い合わせ、その参照を変数 x に代入している。これにより、通信相手を動的に切り替えることができる（詳しくは4節参照）。この部分のπ計算への変換には、以下で述べる通常のメソッド呼び出しをπ計算のプロセス式に変換する手法が適応できる。

t = x.a(); では、x のメソッドを呼び出すので、メソッド呼び出しのπ計算プロセスは

$$X_a = \overline{ap}\,\langle sp, ma\rangle.sp(t)$$

となる。sp は、自身の参照を表す $classB$ 固有のポートである。ma は class A の持つメソッドを指定するためのポートである。ポート sp で受け取ったメソッドの戻り値を t に代入するための入力動作 $sp(t)$ を連接している。フィールド呼び出し処理もメソッド呼び出しと同様に変換することができる。

$$X_{ai} = \overline{ap}\langle sp, fai\rangle.\overline{sp}\,1$$

実際にフィールドに 1 を入力する部分は $\overline{sp}\,1$ である。class B をπ計算プロセス式に変換すると、すべての処理は逐次処理されていくため

$$classB = X_a.X_{ai}$$

と変換できる。

4．HORB を用いた Java プログラムからπ計算への変換

HORB は Java プログラムでリモートオブジェクトを提供しいる。このために、リモートオブジェクトとするクラスの Proxy クラスを自動作成し、リモートなオブジェクトを Proxy を介することによりローカルなオブジェクトとして取り扱うことができる。また、Proxy クラスを呼び出す文は、Java のインスタンスによるメソッド呼び出しと同様の記述方法で記述できる。したがって、リモートなオブジェクトを対象としても HORB を用いることによりπ計算プロセス式への変換が可能となる。図1では、リモートなオブジェクトからインスタンスを受け取る

```
x = sv.broker(ObjA);
```

のような記述がある。sv はリモートなオブジェクトの参照を表し、broker(ObjA) は戻り値として ObjA という名前を持つリモートオブジェクトの参照を返すメソッドである。これにより、参照するリモートオブジェクト x を動的に変更することができる。この処理は、π計算プロセスのポート名の書き換えと同じものだとみなすことができる。このように分散環境で並行に動作するオブジェクト間の動作を HORB を用いて記述することによりその動作をπ計算のプロセス式で簡単に変換することが可能となる。

5．記述例

HORB を用いて分散環境で2人のユーザがチャットを行うプログラムを作成した。作成したこのプログラムは1つの Server クラスのオブジェクトと2つの Client クラスのオブジェクトで構成される。双方の Client が Server にアクセスすることでチャットが開始する。この際、Client-Server 通信が、Client-Client 間通信に切り替わることになるが、これは Java のオブジェクトの参照渡しの機能を利用するものであり、直接通信相手の切り替えを意識しない形で Client が記述できる。また、4節で述べたとおり HORB を用いることで分散環境で実行することができる。

作成した Server クラスは総行数 892 行、総メソッド数 5、総フィールド数 8 となる。Client クラスは総行数 361 行、総メソッド数 17、総フィールド数 17 となった。提案手法を用いて Server クラスをπ計算プロセスに変換すると総動作式数 136、総ポート名数 55、総演算子数 62 程度となった。

6．今後の課題

現在は手作業でπ計算プロセスに変換しているが自動的に変換する処理系の構築し、様々な Java プログラムに適応する予定である。これにより、Java プログラムからπ計算プロセス式への変換の有用性を確認できると考えている。

参考文献

[1] J.Parrow,:An Introduction to the π-Calculus, in J.A.Bergstra,,A.Ponse, and S.A.smolka, eds.,HAND-BOOK OF PROCESS ALGEBRA, NORTH-HOLLAND (2001).

[2] HORB:ネットワークコンピューティングの魔法の絨毯 (http://horb.a02.aist.go.jp/horb-j/)

[3] Björn Victor and Faron Moller,:The Mobility Workbench:A Tools for π-Calculus(1994).