中間発表 恩田晴登 (2022 年 12 月 6 日)

Python の静的型検査器によるコレオグラフィックプログラミング言語の実装

1 初めに

コレオグラフィは，いくつかのロールが互いに連携して何かを行う方法を定義するマルチパーティプロトコルである．本研究の先行研究である Choral[1] は，コレオグラフィのプログラミング用に静的に型付けされた，高レベルの抽象化を備えたオブジェクト指向言語であり，Java を直接拡張したものである．Choral を使用して，分散認証プロトコルや暗号化プロトコルなどをプログラミングすることができる．さらに，Choral コンパイラがコレオグラフィを各ロールのライブラリに自動的に変換（エンドポイント射影）するため，開発者は各ロールのプログラムを記述する必要はなく，変換されたライブラリは通信の安全性が保証されている．本研究では Java を拡張した Choral を参考に，python の静的型検査器である Mypy[2] を使用して，Python をベースとしたコレオグラフィックプログラミング言語の開発を行う．

2 Choral

コレオグラフィックプログラミング言語である Choral について，簡単な例により Choral プログラムから各ロールの Java ライブラリが生成されることを説明する [3]．Choral では，コレオグラフィはオブジェクトとして扱う．Choral のオブジェクトは $T@(R1, R2, \ldots, Rn)$ という形式の型を持つ．ここで T はオブジェクトのインターフェースであり，$R1, R2, \ldots, Rn$ はオブジェクトを共同で実装するロールである．オブジェクトの状態と振舞いは，その型のロールに分散させることができる．

Listing 1: HelloRoles(Choral program)

```
1 class HelloRoles@( A, B ) {
2   public void sayHello() {
3     String@A a = "Hello from A"@A;
4     String@B b = "Hello from B"@B;
5     System@A.out.println( a );
6     System@B.out.println( b );
7   }
8 }
```

上のコードでは，A,B の 2 つのロールをパラメータとするクラスを定義している．メソッド $sayHello()$ では，ロール A における文字列 $String@A$ 型の変数 a を定義し，これに A が持つ "$Hello\ from\ A$" という値を代入している（B も同様である）．最後の 2 行は System オブジェクトを使って変数 a,b を表示している．重要なのは $@R$ という型を持つという点である．Choral ではロールはデータ型の一部であり，例えば

```
1 String@A a = "Hello from B"@B
```

という代入文は左辺と右辺のロールが異なるため型エラーとなる．

Listing 2: HelloRoles(generated Java program)

```
1 class HelloRoles_A {
2   public static void sayHello() {
3     String a = "Hello from A";
4     System.out.println( a );
5   }
6 }
```

Choral で HelloRole クラスが与えられると，コンパイラは Choral のクラスに従って，各ロールに対する振舞いを持つ Java クラスを生成する．上のコードは Choral コンパイラによって自動生成されたロール A の Java クラス HelloRoles_A である．このように Choral は，マルチパーティプロトコルに従った各ロールのプログラムが安全性が保証された状態で自動的に生成されるため，プログラマーの負担を遥かに減らすことができる．

3 Mypy

Choral のエンドポイント射影はプログラムの型情報を使って定義している．しかし，Python は動的言語であり，型検査の機能はないため，エンドポイント射影をする際にプログラムから型情報を得る方法がない．そこで，Python の静的型検査器である Mypy を使い，問題を解消する．Mypy の内部機能により構文解析をして，Visitor パターンを利用して構文木をトラバースすることでコレオグラフィックプログラムから型エラーがない様に，各ロールの Python ライブラリへとエンドポイント射影ができると考えている．

Visitor パターンとは，オブジェクト指向プログラミングにおいて，アルゴリズムをデータ構造から分離するためのデザインパターンの一種である．Visitor パターンでは，データを保持するクラスとアルゴリズムを実装するクラス（Visitor クラス）に分ける．また，それぞれ accept() メソッドと visit() メソッドを持つ．そして，定義したメソッド内で互いのメソッドを呼び出すようにする．本研究では，構文木のトラバース毎に Visitor クラスを作り，コンストラクタ毎に visit() メソッドを加えていく．データ構造は構文木であり，それぞれのコンストラクタに accept() メソッドが用意されている．accept() メソッドは Visitor クラスから対応するメソッドを呼ぶことで，パターンマッチさせる．

4 現在の進捗

Choral の使用例が記載されている論文 [3] を読み，Choral に対しての理解を深めた．また，Mypy を Python プログラムに反映し，型検査を行うことで Mypy に対する理解を深め，Visitor パターンについても学んだ．Python では Choral にある”Hello”@A のような，リテラルにロールの注釈をつける方法が存在しないため，代わりに at(”Hello”, A()) のように at 関数を定義して同じように記述できるようにした（この記法は煩雑なため，今後修正予定）．現在は Choral のエンドポイント射影の中で比較的簡単な代入文を，Mypy を利用した Python ベースのコレオグラフィック言語で記述している途中である．

5　今後の課題

・Choral についての論文をさらに読み，構文やエンドポイント射影の定義に対する理解を深める．
・Visitor パターンを活用し，文や式，クラスなどに対して適切なロールが反映されてエンドポイント射影がされるように考え，プログラムを記述していく．
・Java には存在して Pyhon には存在しないインターフェースを，代わりにどのように記述するかを考え，実行する．
・開発した言語を使って実用例を複数示す．

参考文献

[1] Choral. https://www.choral-lang.org/index.html

[2] mypy. https://mypy.readthedocs.io/en/stable

[3] Object-Oriented Choreographic Programming
SAVERIO GIALLORENZO FABRIZIO MONTESI MARCO PERESSOTTI
https://arxiv.org/abs/2005.09520