Python の静的型検査器を活用した コレオグラフィックプログラミング言語の実装

岐阜大学大学院 自然科学技術研究科 知能理工学専攻　草刈研究室　恩田晴登

1. はじめに

一般的に並列処理を用いたプログラムは，動作の並行性に起因するデッドロックなどのエラーや非決定性問題の発見と修正が困難であるため構築が難しい．コレオグラフィックプログラミング言語は，デッドロック等の並行性起因エラーが起こらないことが保証されているプログラミング言語である．本研究では，先行研究である Java を拡張したコレオグラフィックプログラミング言語 Choral[1] の理論から，機械学習や IoT の業界で盛んに使用されている Python ベースのコレオグラフィックプログラミング言語である PyChoral の実装を，Python の静的型検査器である Mypy を活用して行い，有用性を示す．

2. コレオグラフィとエンドポイント射影

コレオグラフィとは，並行に動作する複数の参加者の連携手順をまとめたプログラムである．これに従って各エンドポイント (通信の参加者) のプログラムが生成され，それらはデッドロック等の並行性起因のエラーがないことを保証する．コレオグラフィに従った各エンドポイントのプログラムはエンドポイント射影 [2] という動作により，自動的に抽出される．エンドポイント射影とは，コレオグラフィから各参加者の型情報を導出する操作である．

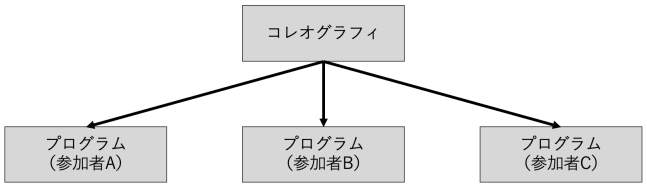

図 1　エンドポイント射影 (参加者 3 人)

3. PyChoral

PyChoral は Python ベースのコレオグラフィックプログラミング言語である．Python は動的型付け言語であるため，Python の静的型検査器である Mypy を使用してコンパイル時に型検査を行う．PyChoral でコレオグラフィを記述し，コンパイル時に各参加者の Python プログラムがコレオグラフィに従った形で自動導出する．code1 は PyChoral プログラムの例である．

code 1　Staff(S) と Customer(C) でのお会計時のやり取り

```
1  class Check(Ch2[S,C]):
2    def __init__(self):
3      self.ch_CS:Channel[int,C,S] = Channel[int,C,S
           ]('C','S')
4      self.ch_SC:Channel[str,S,C] = Channel[str,S,C
           ]('S','C')
5    def check(self,price:At[int,S],money:At[int,C])
         -> At[str,C]:
6      payment = self.ch_CS.com(money)
7      if (payment > price):
8        return self.ch_SC.com("thanks"@S())
9      else:
10       return self.ch_SC.com("not enough"@S())
```

クラス Check の引数である Ch2[S,C] は，参加者 S,C がこのクラスに関連することを表す．Channel は第 1 引数に通信する値の型，第 2,3 引数にそれぞれ送信者、受信者の型をもつ型クラスである．At は第 1 引数に値の型，第 2 引数にその値に関連する参加者の型をもつ型クラスである．e@R() は参加者 R の値 e であることを示す．com は送信者の値を受信者に転送するメソッドである．

PyChoral は図 2 のように，一度型検査を行なった後に各エンドポイントの AST を再構築し，各エンドポイントの Python プログラムを生成する．

図 2　PyChoral プロセス

3.1. PyChoral の仕様

図 3 は PyChoral の構文のうち，既存の Python との差分がある部分である．クラス定義は，参加者の明示をするために Ch クラスを追加し，これを第一引数にとるものとする．関数定義は，引数 (self 以外) に型注釈をつけることで関連する参加者を示す．式はリテラルのみ @ をつけて参加者の情報を得る．その他の式や文は型注釈もしくは Mypy による型検査の際に取得できる型情報から参加者の情報を得る．

Class	$Class$	$::= \mathsf{class}\ id\ (\mathrm{Ch}[\overline{A}], \overline{Exp})$
Function	$Func$	$::= \mathsf{def}\ id\ (\overline{id(:TE)})$
Annotation	AN	$::= @id$
Expression	Exp	$::= lit@\overline{A()} \mid f(\overline{Exp}) \mid Exp.f(\overline{Exp}) \mid \cdots$

図 3　PyChoral の構文 (一部)

とあるコレオグラフィの参加者 A に対する PyChoral の項の射影は $(\!|Term|\!)^A$ と記し，これは $Term$ における参加者 A の振舞いを表す Python の項となる．例えば，クラス定義の射影はクラス継承の第一引数 Ch クラスに存在する

参加者名によって Python の項を生成する．

$$(\!|\text{class } id\ (Ch[\overline{R}], \overline{Exp}) : \overline{Stm}|\!)^A = \begin{cases} \text{class } id\_A\ (\overline{(\!|Exp|\!)^A}) : \overline{(\!|Stm|\!)^A} & \text{if } A \in \overline{R} \\ \text{absent} & \text{if } A \notin \overline{R} \end{cases}$$

(例) 参加者 A, B が関わるクラス Foo の射影

PyChoral	⇒	Python
$(\!|$class Foo(Ch2[A,B])$|\!)^A$	⇒	class Foo_A( )
$(\!|$class Foo(Ch2[A,B])$|\!)^B$	⇒	class Foo_B( )

3.2. **エンドポイント射影の実装**　本研究では既存の Python パーサーを活用する．Choral では**@R** を各インターフェースに付属させることで参加者の情報を取得していたが，Mypy の検査器は**@**を認識できないため，その際に参照する標準ライブラリ typeshed 内のファイルに **At** クラス，**Channel** クラスを追加し，参加者の情報を取る．

```
class At(Generic[T1,R1],T1):
  pass
class Channel(Generic[T1,R1,R2]):
  def com(self,msg:At[T1,R1]) -> At[T1,R2]:
    pass
  def select(self,msg:At[T1,R1]) -> At[T1,R2]:
    pass
class object:
  ...
  def __matmul__(self:Self, _:R1) -> At[Self,R1]:
      ...
```

_T1，_R1，_R2 はジェネリック型であり，任意の型をとる．通信メソッド com，選択メソッド select は，Channel の第 2 引数から第 3 引数へ参加者の情報が追加された値を転送するメソッドとして定義する．

PyChoral プログラムの射影はトップレベルの項から行われる．projection は Statement のリスト n，射影される参加者 r，型検査器 tc を引数にとり，新しいデータ構造を返す関数である．n の要素 node に対して isinstance() で型を制限し，それぞれの射影関数に渡す．

code 2　projection.py

```
def projection_all(n:list[Statement],r:str,tc:
    TypeChecker) -> list[Stmt]:
  result:list[Stmt] = [ ]
  for node in n:
      if isinstance(node,ClassDef):
          result += [projection_class(node,r,tc)]
      ...
  return result
```

Import を含む文，ブロック，クラス定義，関数定義は射影後に独自のデータ構造で生成される．

code 3　data.py

```
class Stmt:
    pass
class Asg(Stmt): #Assignment
  def __init__(self, l:str, r:str, t:Type|None):
      self.lvalues = l
      self.rvalue = r
      self.type = t
...
def stmt_to_string(s:Stmt,indent:int) -> str:
  if isinstance(s,Asg):
    return " "*indent+s.lvalues+" = "+s.rvalue
  ...
```

親クラス Stmt は抽象的な構文木であり，クラス継承を使って AST の具体的なノードを子クラス (Asg,ClassDef,FuncDef,...) として定義している．新しく定義したデータクラスは Mypy に備わっている標準のデータクラスから射影定義に従って必要なパラメータのみ抽出したものである．射影により生成されたデータクラスはマージや正規化で処理した後に，関数 stmt_to_string により Python ファイルに文字列として出力される．

3.3. **アプリケーションによる評価**

4. 結論

本研究は Python を拡張したコレオグラフィックプログラミング言語 PyChoral を実装した．PyChoral プログラムは射影の定義によって各参加者の Python プログラムが生成される．これらはコレオグラフィに従っているため，デッドロック等の並行性起因エラーが生じないことが保証されている．これにより，マルチスレッド環境におけるプログラムが単一の言語で記述されるプログラムによって実装可能となり，Python プログラマの手助けとなる．3 者間までの通信しか対応していないため，それ以上の参加者に対応可能にすることが今後の課題である．

参考文献

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

[2] Alceste Scalas and Nobuko Yoshida. Less is more: multiparty session types revisited. *POPL*, 2019.