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

機械学習や IoT の業界で盛んに使用されている Python は GIL がオプション化されるため，並列処理プログラミングの需要が高まると考える．しかし，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．この問題の解決手法の一つとして，コレオグラフィがある．

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り，コレオグラフィックプログラミング言語によって記述する．コレオグラフィ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] 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．先行研究である Choral[1] は，Java を拡張したコレオグラフィックプログラミング言語の一つであり，エンドポイント射影によって並行性起因のエラーなく実行可能である各エンドポイントの Java プログラムが生成される．

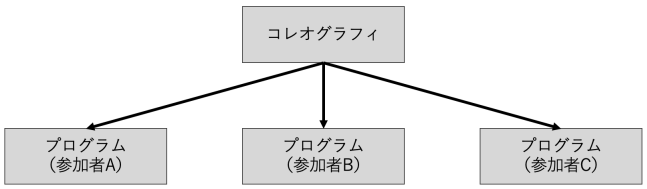

図 1　エンドポイント射影 (参加者 3 人)

本研究では，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. 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，Mypy を使用して静的に型検査する．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 で記述したコレオグラフィの例である．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 は転送される値，送信者，受信者の型をもつ型クラスである．At は転送される値，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．

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

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

この PyChoral で記述したコレオグラフィより各参加者 (S,C) の Python プログラムを自動導出する．

```
1 class Check_C():
2   def __init__(self):
3     self.ch_CS = Channel[int,C,S]()
4     self.ch_SC = Channel[str,S,C]()
5   def check(self,money):
6     Unit.id(self.ch_CS.com(money))
7     return self.ch_SC.com()
```

code 2　生成された Python プログラム (Customer)

生成された Python プログラムは型注釈がない．PyChoral プログラムの型注釈において射影対象でない参加者が関連する値は消える．射影の実装の詳細は修士論文に掲載する．

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．

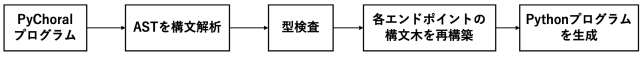

図 2　PyChoral プロセス

2.1. PyChoral の構文・Python への射影の定義

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

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( )

2.2. **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 内に **At** クラス，**Channel** クラスを追加し，参加者の情報を取る．

```
1 class At(Generic[T1,R1],T1):
2   pass
3 class Channel(Generic[T1,R1,R2]):
4   def com(self,msg:At[T1,R1]) -> At[T1,R2]:
5     pass
6   def select(self,msg:At[T1,R1]) -> At[T1,R2]:
7     pass
8 class object:
9   ...
10  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．また，object クラスに@のためのメソッド__matmul__を追加する．これにより，Mypy の型検査で@が型エラーにならない．

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 は文のリスト n, 射影対象の参加者 r, 型チェッカー tc を引数にとり，新しいデータ構造を返す関数である．各文，各式は Python の標準関数 isinstance() で型を制限し，再帰的に射影を行う．isinstance() はオブジェクトが指定されたクラスまたは型に属しているかどうかを判定する関数である．

```
1 def projection_all(n:list[Statement],r:str,tc:
      TypeChecker) -> list[Stmt]:
2   result:list[Stmt] = [ ]
3   for node in n:
4       if isinstance(node,ClassDef):
5           result += [projection_class(node,r,tc)]
6       ...
7   return result
```

code 3　projection.py

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．親クラス Stmt は抽象的な構文木であり，クラス継承により AST の具体的なノードを子クラス (Asg,ClassDef,FuncDef,...) として定義する．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．

```
1 class Stmt:
2     pass
3 class Asg(Stmt): Assignment
4   def __init__(self, l:str, r:str, t:Type|None
        ):
5       self.lvalues = l
6       self.rvalue = r
7       self.type = t
8 ...
9 def stmt_to_string(s:Stmt,indent:int) -> str:
10  if isinstance(s,Asg):
11    return " "*indent+s.lvalues+" = "+s.rvalue
12  ...
```

code 4　data.py

2.3. **アプリケーションによる評価**

3. まとめ

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．PyChoral は，現段階で 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.