2024 年度 卒業/修士論文 (中間発表)

mypy による静的型検査を活用した Python ベースのコレオグラフィック言語の実装

岐阜大学大学院工学研究科 応用情報学専攻 草刈研究室

1224525023　恩田晴登

指導教員　草刈圭一朗

2023 年 12 月 19 日

目次

第 1 章　はじめに

Python は機械学習や IoT の業界で盛んに使用されているプログラミング言語の一つである。Python は現在、グローバル・インタプリタ・ロック (GIL) が存在する。これは、一度に 1 つのスレッドだけが Python バイトコードを実行することを保証するために CPython インタプリタによって使用されるメカニズムである。これにより、オブジェクトモデルが同時アクセスに対して暗黙的に安全になり、インタプリタのマルチスレッド化が容易になるが、弊害としてマルチプロセッサマシンによってもたらされる並列処理の多くが犠牲になる。

しかし、近い将来 Python は GIL の除去可能性があることを発表した。その影響により Python ユーザーがマルチスレッド環境でのプログラミングをすることが増えるであろう。ただ、一般的に並列処理を用いたプログラムはデッドロックなどのエラーを出さずに記述することは困難であるため、バグが大量に出てくる恐れがある。よって、単一のプログラムとしてマルチパーティなプロトコルを記述し、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コレオグラフィック言語があれば、多くの Python プログラマーの不安や負担が解消される。

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 の理論を参考に、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 の実装を行い、マルチスレッドプログラムを Python で書くプログラマにとっての有用性を示す。

第 2 章　コレオグラフィックプログラミング

本研究で実装する PyChoral はコレオグラフィックプログラミング言語の一つである。この章では本研究に関する前提知識と先行研究について述べる。

2.1　コレオグラフィーとエンドポイント射影

コレオグラフィーとは、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パラダイムである。これにより、デッドロック等の並行性起因のエラーが起こる恐れを排除できる。コレオグラフィーに従った各エンドポイント (通信の参加者) の型情報はエンドポイント射影という動作により、自動的に抽出される。

図 2.1　コレオグラフィーとエンドポイント (Alice,Bob,Carol の会話)

ただし、通常のコレオグラフィーはコンパイル時にエンドポイントの型情報を導出するだけであり、そこからプログラマーが手動で導出した型情報を元に各エンドポイントのプログラムを記述しなければならないため、バグが起きた際の処理などがプログラマーにとって負担になってしまうという難点がある。

エンドポイント射影とは、コレオグラフィーから各参加者の型情報を導出する操作である。コレオグラフィックプログラミング言語にはコンパイラが付属しており、コンパイラはエンドポイント射影理論によって同時分散システム用の実行可能コードに変換する。

図 2.2　エンドポイント射影 (参加者 3 人)

2.2　Choral

Choral は Java 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である。前節で述べたように、マルチスレッド環境で動作するプログラムでは、データ競合が発生しないようにするのがプログラマの責任であり大きな負担となっていたが、Choral は、分散システムに従わせたいプロトコル全体を単一のプログラムとして作成できるため、これらの負担が軽減される。Choral のオブジェクトには $t@(R_1, \cdots, R_n)$ という形式の型があり、Java の基本的なオブジェクトインターフェイス T に各参加者の情報となるパラメータ $R_1, \cdots, R_n$ が存在する。これにより、Choral プログラムをコンパイルする際に各参加者のプログラムが、コレオグラフィーに従った形で自動的に生成することができる。

以下は Choral のプログラム例と Choral コンパイラによって自動導出される Java プログラムの一例である。

プログラム 2.1　Choral プログラムの例 (分散認証)

```
1  enum AuthBranch { Ok,KO }
2
3  public class DistAuth@(Client,Service,IP){
4    private TLSChannel@(Client,IP)<object> ch_Client_IP; #あ aaa
5    private TLSChannel@(Service,IP)<object> ch_Service_IP;
6    ...
7    public AuthResult@(Client,Service) authenticate(Credentials@Client credentials) {
8      String@Client salt = ch_Client_IP.<String>com(ClientRegistry@IP.getSalt(
           ch_Client_IP.<String>com(credentials.username)));
9      Boolean@IP valid = ClientRegistry@IP.check(ch_Client_IP.<String>com(calcHash(
           salt, credentials.password)));
10     if (valid) {
11       ch_Client_IP.<AuthBranch>select(AuthBranch@IP.OK);
12       ch_Service_IP.<AuthBranch>select(AuthBranch@IP.OK);
13       AuthToken@IP t = AuthToken@IP.create();
14       return new AuthResult@(Client,Service)(
15       ch_Client_IP.<AuthToken>com(t), ch_Service_IP.<AuthToken>com(t)
16       );
17     } else {
18       ch_Client_IP.<AuthBranch>select(AuthBranch@IP.KO);
19       ch_Service_IP.<AuthBranch>select(AuthBranch@IP.KO);
20       return new AuthResult@(Client,Service)();
21     }
22   }
23 }
```

2.1 は Client,Service,IP の 3 つの役割からなる分散認証システムを簡約したものである。Choral プログラムはクラス名や変数、型などに @ でロール名の情報を加えている。また、Choral には Java にはない com メソッドと selection メソッドなるものがある。

プログラム 2.2　データ転送のための基本的な有向チャンネル

```
1 interface DiDataChannel@(A,B)<T@C> {
2   <S@D extends T@D> S@B com(S@A m);
3 }
```

2.2 は com メソッドの最も簡易的に使用しているインターフェースの例である。DiDataChannel は、A と B で抽象化された 2 つのロール間の有向チャンネルで、型パラメータ T で抽象化された型のデータを A から B に転送するためのインターフェースである。データ転送は com を呼び出すことによって実行される。com は、A に位置する T のサブタイプの任意の値 S@A を取り、S@B を返す。例えば 2.1 の 8 行目の外側の com は Client から IP へ String 型のメッセージ (credentional@IP.getsalt(...)) を転送することを意味する。

プログラム 2.3　selection メソッドの定義

```
1 interface DiSelectChannel@(A,B) {
2   @SelectionMethod
3   <T@C extends Enum@C<T>> T@B select(T@A m);
4 }
```

selection メソッドはロール間で列挙型のインスタンスを送信する際に使用する。例えば、2.1 の 11 行目、18 行目では Client から IP へ列挙型の値 OK,KO を転送することを意味する。selection メソッドは選択を意味し、プロジェクション後は switch 文として Java プログラムに現れる (4.2)。

プログラム 2.4　生成された Client の Java プログラム (分散認証)

```
1 public class DistAuth_Client {
2   private TLSChannel_A<Object> ch_Client_IP;
3
4   public DistAuth_Client(TLSChannel_A <Object> ch_Client_IP) {
5     this.ch_Client_IP = ch_Client_IP;
6   }
7
8   private String calcHash( String salt, String pwd ) { /*...*/ }
9
10  public AuthResult_A authenticate(Credentials credentials) {
11    String salt = ch_Client_IP.<String>com(ch_Client_IP.<String>com(credentials.
          username));
12    ch_Client_IP.<String>com(calcHash(salt, credentials.password));
13    switch (ch_Client_IP.<AuthBranch>select(Unit.id)) {
14      case OK -> { return new AuthResult_A( ch_Client_IP.<AuthToken>com(Unit.id),
            Unit.id); }
15      case KO -> { return new AuthResult_A(); }
16      default -> { throw new RuntimeException( /*...*/ ); }
17    }
18  }
19 }
```

Client,Service,IP のうち、Client にプロジェクションすると 4.2 が生成される。Java プログラムには@が存在せず、Choral プログラムで Client が関係する式や文のみ射影後に残る。2.1 の 10〜21 行目の if

文は 4.2 の 14〜17 行目の switch 文に変換されており、case は列挙型の値で場合分けされて、それぞれ if 節と else 節の return 文をもつ。

このように Choral プログラムからコンパイラーを通して各エンドポイントの Java プログラムが自動導出される。それぞれの Java プログラムはコレオグラフィーに従っているため、相互関係によるバグがないことが保証されている。これは、マルチパーティなプロトコルを記述する Java プログラマーにとっては大きなメリットである。

第 3 章　mypy

Python は動的型付け言語で実行時にのみエラーが表示されるのでコンパイル時に型に対するエラーは表示されない。mypy は Python の静的型検査器であり、既存の Python コードに型アノテーションを追加することで、型が誤っていると警告を出すようになる。これによりコンパイル時にバグの検出が可能になり、安全なコーディングが可能となる。

プログラム 3.1　型注釈のない Python コード

```
1 def greeting(name):
2   return 'Hello ' + name
3
4 greeting(123)
5 greeting(b"Alice")
```

プログラム 3.2　型注釈のある Python コード

```
1 def greeting(name: str) -> str:
2   return 'Hello ' + name
3
4 greeting("World!") # No error
5 greeting(3) # Argument 1 to "greeting" has incompatible type "int"; expected "str"
6 greeting(b'Alice')
7 # Argument 1 to "greeting" has incompatible type "bytes"; expected "str"
```

本研究では mypy を別の Python アプリケーションに統合するために、Python プログラムに mypy.api をインポートして型検査を行う。以下は通常の mypy を使用した際の型検査のプロセスである。

図 3.1　mypy プロセス

第 4 章　PyChoral

4.1　PyChoral の概要

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。PyChoral でマルチパーティなプログラムを記述し、コンパイルすることで型エラーがない場合は、各参加者の Python プログラムコードがコレオグラフィーに従った形で自動導出することができる。

PyChoral では mypy を活用した型検査のプロセスを改造し、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 4.1　PyChoral プロセス

PyChoral は Python ベースの言語のため、Java ベースの Choral を模倣するには静的型付け言語として扱え、コンパイル時にエラーの有無を判別したいため Mypy を活用する。ただし、Mypy で型注釈をつけられれば Choral と同じように動くかというと、そうではない。

まず、Choral は Java のパーサーを改良し、独自のコンパイラーを使用して Choral プログラムから Java プログラムを自動導出しているが、本研究では既存の Python パーサーをそのまま活用できるようにした。

Choral を真似するだけではだめ！ Choral との違い、苦労した点などを書く。

- mypytest でのプロジェクションの仕方
- @をどうしたか
- At クラス追加

4.2　PyChoral によるプログラム例

Choral の AuthDist を参照した。（まだ実際はできていない）

プログラムの説明をここに書く。

プログラム 4.1　PyChoral プログラムの例 (分散認証)

```
1 class AuthBranch(Ch1[IP],Enum):
2   OK = "OK"
3   KO = "KO"
4
5 class DistAuth(Ch3[Client,Service,IP]):
6   def __init__(self,ch_Client_IP:Channel[object,Client,IP],ch_Service_IP:Channel[
        object,Service,IP]):
7     self.ch_Client_IP = ch_Client_IP
8     self.ch_Service_IP = ch_Service_IP
```

```
9   ...
10  def authenticate(self,credentials:At[Credentials,Client]) -> Channel[AuthResult,
        Client,Service]:
11    salt:At[str,Client] = ch_Client_IP.com(ClientRegistry.get_salt(ch_Client_IP.com(
          credentials.username)))
12    valid:At[boolean,IP] = ClientRegistry.check(ch_Client_IP.com(calcHash(salt,
          credentials.password)))
13    if valid:
14      ch_Client_IP.select(AuthBranch.OK@IP())
15      ch_Service_IP.select(AuthBranch.OK@IP())
16      t:At[AuthToken,IP] = AuthToken.create()
17      return AuthResult[Client,Service](ch_Client_IP.com(t):Channel[AuthResult,
            Client,Service], ch_Service_IP.com(t):Channel[AuthResult,Client,Service])
18    else:
19      ch_Client_IP.select(AuthBranch.KO@IP())
20      ch_Service_IP.select(AuthBranch.KO@IP())
21      return AuthResult[Client,Service]()
```

プログラム 4.2　生成された Client の Python プログラム例 (分散認証)

```
1  public class DistAuth_Client():
2    def __init__(self,ch_Client_IP):
3      self.ch_Client_IP = ch_Client_IP
4    ...
5    def calcHash(self,salt,pwd):
6      ...
7    def authenticate(self,credentials):
8      salt = ch_Client_IP.com(ch_Client_IP.com(credentials.username))
9      ch_Client_IP.com(calcHash(salt,credentials.password))
10     match ch_Client_IP.select(Unit.id):
11       case OK:
12         return (ch_Client_IP.com(Unit.id),Unit.id)
13       case KO:
14         return ()
15       case __:
16         raise Exception
```

4.3　PyChoral の仕様

この節では本研究で実装した PyChoral プログラムの構文及びコンパイル時のエンドポイントプロジェクションの定義を示す。まず、PyChoral の構文を示し、Python との違いについて説明する。次に、PyChoral プログラムをコンパイラーによって Python プログラムが自動導出されるためのエンドポイントプロジェクションの定義を示す。

4.3.1 syntax

4.3.2 merging

4.3.3 normalizing

4.4 実装

- typeshed の改造
- pro_s, pro_e, pro_class, pro_func

第 5 章　今後の課題

付録

Projection to Python

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

$$(Func)\quad (\!|\textsf{def } id\ (\overline{id : TE}) : \overline{Stm}|\!)^A = \textsf{def } id\ (\overline{id}) : \overline{(\!|Stm|\!)^A}$$

$$(Exp)\quad (\!|lit@(\overline{B}) : \tau|\!)^A = \begin{cases} lit & \text{if } A \in \overline{B} \\ \texttt{Unit.id} & \text{otherwise} \end{cases}$$

$$(\!|Exp.id : \tau|\!)^A = \begin{cases} (\!|Exp|\!)^A.id & \text{if } A \in \text{rolesOf}(Exp.id) \\ \text{absent} & \text{otherwise} \end{cases}$$

$$(\!|f(\overline{Exp}) : \tau|\!)^A = \begin{cases} f(\overline{(\!|Exp|\!)^A}) & \text{if } A \in \text{rolesOf}(f(\overline{Exp})) \\ \texttt{Unit.id}(f(\overline{(\!|Exp|\!)^A})) & \text{if } A \in \text{rolesOf}(\overline{Exp}) \wedge A \notin \text{rolesOf}(f(\overline{Exp})) \\ \texttt{Unit.id}(\overline{(\!|Exp|\!)^A}) & \text{otherwise} \end{cases}$$

$$(\!|Exp.f(\overline{Exp}) : \tau|\!)^A = \begin{cases} (\!|Exp|\!)^A.f(\overline{(\!|Exp|\!)^A}) \\ \qquad \text{if } A \in \text{rolesOf}(Exp) \wedge A \in \text{rolesOf}(\overline{Exp}) \\ \qquad\qquad \wedge A \in \text{rolesOf}(Exp.f(\overline{Exp})) \\ \texttt{Unit.id}((\!|Exp|\!)^A.f(\overline{(\!|Exp|\!)^A})) \\ \qquad \text{if } A \in \text{rolesOf}(Exp) \wedge A \notin \text{rolesOf}(Exp.f(\overline{Exp})) \\ \texttt{Unit.id}((\!|Exp|\!)^A, \overline{(\!|Exp|\!)^A}) \quad \text{otherwise} \end{cases}$$

$$(\!|C[\overline{B}](\overline{Exp}) : \tau|\!)^A = \begin{cases} (\!|C[\overline{B}]|\!)^A(\overline{(\!|Exp|\!)^A}) & A \in \overline{B} \\ \texttt{Unit.id}(\overline{(\!|Exp|\!)^A}) & \text{otherwise} \end{cases}$$

$$(\!|Exp_1\ BinOp\ Exp_2|\!)^A = (\!|Exp_1|\!)^A\ BinOp\ (\!|Exp_2|\!)^A$$

$$\text{rolesOf}(\_ : \tau@(\overline{B})) = \overline{B}$$

$$\text{rolesOf}(\overline{Exp}) = \bigcup_i \text{rolesOf}(Exp_i)$$

$$\overline{(\!|Exp|\!)^A} = Exp'_1, Exp'_2, \cdots, Exp'_n \quad \text{where} \quad Exp'_i = (\!|Exp_i|\!)^A$$

$$(Stm)\quad (\!|\textsf{pass}|\!)^A = \textsf{pass}$$

$$(\!|\textsf{return } Exp;|\!)^A = \textsf{return } (\!|Exp|\!)^A$$

$$(\!|Exp; \overline{Stm}|\!)^A = \begin{cases} \textsf{match } (\!|Exp|\!)^A : \\ \quad \textsf{case } id_2 : \overline{(\!|Stm|\!)^A}; & \text{if } Exp = Exp'.\texttt{select}(id_1@A.id_2) : \texttt{Enum@}A \\ \quad \textsf{case } \_ : \text{assert False}; \\ (\!|Exp|\!)^A; \overline{(\!|Stm|\!)^A} & \text{otherwise} \end{cases}$$

$$(\!|id : TE = Exp\ ; \overline{Stm}|\!)^A = \begin{cases} id = (\!|Exp|\!)^A; \overline{(\!|Stm|\!)^A} & \text{if } A \in \text{rolesOf}(TE) \\ (\!|Exp|\!)^A; \overline{(\!|Stm|\!)^A} & \text{otherwise} \end{cases}$$

$$(\!|Exp_1\ AsgOp\ Exp_2\ ; \overline{Stm}|\!)^A = (\!|Exp_1|\!)^A AsgOp\ (\!|Exp_2|\!)^A\ ; (\!|\overline{Stm}|\!)^A$$

$$(\!|\textsf{if}\ Exp : Stm_1\ ; \textsf{else} : Stm_2\ ; \overline{Stm}|\!)^A =$$

$$\begin{cases} \textsf{if}\ (\!|Exp|\!)^A : (\!|Stm_1|\!)^A\ ; \textsf{else} : (\!|Stm_2|\!)^A\ ; (\!|\overline{Stm}|\!)^A & \text{if rolesOf}(Exp) = A \\ (\!|Exp|\!)^A\ ; [\![(\!|Stm_1|\!)^A]\!] \sqcup [\![(\!|Stm_2|\!)^A]\!]\ ; (\!|\overline{Stm}|\!)^A & \text{otherwise} \end{cases}$$

$$(\!|\textsf{raise}\ Exp|\!)^A = \textsf{raise}\ (\!|Exp|\!)^A$$

$$(\!|\textsf{assert}\ Exp|\!)^A = \textsf{assert}\ (\!|Exp|\!)^A$$

$$(\!|\overline{Stm}|\!)^A = Stm'_1, Stm'_2, \cdots, Stm'_n \quad \text{where} \quad Stm'_i = (\!|Stm_i|\!)^A$$

Merging

Statement

$$\dot{\sqcup}\ \overline{Stm} = \dot{\sqcup}\ (Stm_1, \cdots, Stm_n) = [\![Stm_1]\!] \sqcup \cdots \sqcup [\![Stm_n]\!]$$

$$\textsf{return}\ Exp \sqcup \textsf{return}\ Exp' = \textsf{return}\ Exp \sqcup Exp'$$

$$\textsf{raise}\ Exp \sqcup \textsf{raise}\ Exp' = \textsf{raise}\ Exp \sqcup Exp'$$

$$(Exp_1\ AsgOp\ Exp_2; \overline{Stm}) \sqcup (Exp'_1\ AsgOp\ Exp'_2; \overline{Stm'})$$
$$= (Exp_1 \sqcup Exp'_1)\ AsgOp\ (Exp_2 \sqcup Exp'_2); (\overline{Stm} \sqcup \overline{Stm'})$$

$$(Exp; \overline{Stm}) \sqcup (Exp'; \overline{Stm'}) = (Exp \sqcup Exp'); (\overline{Stm} \sqcup \overline{Stm'})$$

$$\begin{array}{l} \textsf{if}\ Exp_1 : \\ \quad Stm_1; \\ \cdots \\ \textsf{elif}\ Exp_n : \\ \quad Stm_n; \\ \textsf{else} : \\ \quad Stm_e; \\ \overline{Stm} \end{array} \sqcup \begin{array}{l} \textsf{if}\ Exp'_1 : \\ \quad Stm'_1 \\ \cdots \\ \textsf{elif}\ Exp'_n : \\ \quad Stm'_n \\ \textsf{else} : \\ \quad Stm'_e \\ \overline{Stm'} \end{array} = \begin{array}{l} \textsf{if}\ Exp_1 \sqcup Exp'_1 : \\ \quad Stm_1 \sqcup Stm'_1 \\ \cdots \\ \textsf{elif}\ Exp_n \sqcup Exp'_n : \\ \quad Stm_n \sqcup Stm'_n \\ \textsf{else} : \\ \quad Stm_e \sqcup Stm'_e \\ \overline{Stm} \sqcup \overline{Stm'} \end{array}$$

$$\begin{array}{l} \textsf{match}\ Exp : \\ \quad \textsf{case}\ id_a : Stm'_a; \\ \quad \cdots \\ \quad \textsf{case}\ id_x : Stm'_x; \\ \quad \textsf{case}\ id_y : Stm'_y; \\ \\ \quad \textsf{case}\ \_ : Stm'_{ex}; \\ \overline{Stm} \end{array} \sqcup \begin{array}{l} \textsf{match}\ Exp' : \\ \quad \textsf{case}\ id_a : Stm''_a; \\ \quad \cdots \\ \quad \textsf{case}\ id_x : Stm''_x; \\ \\ \quad \textsf{case}\ id_z : Stm'_z; \\ \quad \textsf{case}\ \_ : Stm''_{ex}; \\ \overline{Stm'} \end{array} = \begin{array}{l} \textsf{match}\ Exp \sqcup Exp' : \\ \quad \textsf{case}\ id_a : Stm'_a \sqcup Stm''_a; \\ \quad \cdots \\ \quad \textsf{case}\ id_x : Stm'_x \sqcup Stm''_x; \\ \quad \textsf{case}\ id_y : Stm'_y; \\ \quad \textsf{case}\ id_z : Stm'_z; \\ \quad \textsf{case}\ \_ : Stm'_{ex} \sqcup Stm''_{ex}; \\ \overline{Stm} \sqcup \overline{Stm'} \end{array}$$

$$\overline{Stm} \sqcup \overline{Stm'} = Stm_1 \sqcup Stm'_1\ ,\ Stm_2 \sqcup Stm'_2\ ,\ \cdots, Stm_n \sqcup Stm'_n$$

Expression

$$Exp \sqcup Exp' = \begin{cases} Exp & \text{if } Exp = Exp' \\ \text{error} & \text{if } Exp \neq Exp' \end{cases}$$

normalizer

Statements

$[\![\textsf{pass}]\!] = \textsf{pass}$

$[\![\textsf{return } Exp]\!] = \textsf{return } [\![Exp]\!]$

$$\textsc{noop}(Exp) = \begin{cases} [blank] & \text{if } Exp \in \{\texttt{Unit.id, None}\} \\ Exp & \text{otherwise} \end{cases}$$

$$[\![Exp_1\ AsgOp\ Exp_2; \overline{Stm}]\!] = \begin{cases} [\![Exp_1]\!]; [\![\overline{Stm}]\!] & \text{if } \textsc{noop}([\![Exp_2]\!]) = [blank] \\ [\![Exp_2]\!]; [\![\overline{Stm}]\!] & \text{if } \textsc{noop}([\![Exp_1]\!]) = [blank] \\ [\![\overline{Stm}]\!] & \text{if } \textsc{noop}([\![Exp_1]\!], [\![Exp_2]\!]) = [blank] \\ [\![Exp_1]\!] AsgOp [\![Exp_2]\!]; [\![\overline{Stm}]\!] & \text{otherwise} \end{cases}$$

$$[\![Exp; \overline{Stm}]\!] = \begin{cases} [\![\overline{Stm}]\!] & \text{if } \textsc{noop}([\![Exp]\!]) = [blank] \\ [\![Exp]\!]; [\![\overline{Stm}]\!] & \text{otherwise} \end{cases}$$

Expressions

$[\![None]\!] = None \quad [\![id]\!] = id$