SRA の技術情報誌
GS LetterNeo
Vol.197　2025 年 3 月

Python で始める論理プログラミング -SWI-Prolog 編-

熊澤 努

技術本部 先端技術研究室

はじめに

Vol.195 に引き続き，今回も Python で論理プログラミングに挑戦していきましょう。Vol.195 では Datalog を使いましたが，今回は Datalog の記述能力を含んでいて，広範な関係性の記述のもとで論理的推論を行うことのできる論理プログラミング言語 Prolog を，Python の処理系で動かします．この記事で紹介するプログラムは，Google Colaboratory の 2025 年 2 月 26 日現在での最新の CPU 環境で動作確認をしています．また，今回の記事の執筆にあたり，以下の記事の例題を引用させていただきました．

https://colab.research.google.com/github/sut-ai/supplementary/blob/master/notebooks/logic_programming/index.ipynb#scrollTo=b530eaa2

SWI-Prolog を準備する

2025 年 2 月現在，最も有名な Prolog の処理系の一つは SWI-Prolog[1]でしょう．SWI-Prolog は Windows や MacOS 上で動かすことができます． SWI-Prolog は各計算機環境にインストールをして使うことが多いと思いますが，Google Colaboratory（以下，Google Colab）の Python 実行環境上で SWI-Prolog を動かすこともできます．ここからは，Google Colab のコード記述用のセルの内容を示しながら説明していきます．

まず，Google Colab の CPU 実行環境を起動します．次に，SWI-Prolog を実行環境上にインストールします．SWI-Prolog は apt-get コマンドでインストールすることができます。

```
!apt-get install swi-prolog
```

次に，Python で SWI-Prolog を使うためのインタフェースを提供するパッケージ pyswip[2]をインストールします．

```
!pip install pyswip
```

以上で準備ができました．

簡単な親子関係の計算

SWI-Prolog を使うためには pyswip から必要なモジュールをインポートしておきます．

```
from pyswip import Prolog
```

それでは，実際に Prolog のプログラムを書いてみます．例として，「マイケルはジェーンの父親である」と，「Y が X の父親ならば，X は Y の子供である」という親子の関係性を記述すると，以下のようになります．

```
prolog = Prolog()
prolog.assertz("father(michael,jane)")
prolog.assertz("child(X,Y) :- father(Y,X)")
```

Python プログラムで Prolog クラスのインスタンスを生成し，prolog.assertz を繰り返し読んで，Prolog のプログラムを順に追加していきます．pyDatalog とは異なり，Prolog プログラム自体は文字列として記述します．Prolog は定数は小文字始まり，変数は大文字始まりです．したがって，michael と jane は定数，X と Y は変数です．文字列 :- は含意（「ならば」）

[1] https://www.swi-prolog.org/
[2] https://pyswip.org/

を表す記号です．含意は，左から右ではなく，右から左に読むことに注意しましょう．上のプログラムでは「child(X,Y)ならば father(Y,X)」と読むのではなく，「father(Y,X)ならばchild(X,Y)」と読みます．また，Prolog には文末をピリオド . で終えるという構文上の規則がありますが，pyswip で文末にピリオドを付けるとエラーになるので注意が必要です．

次に問い合わせを記述します．pyDatalog のときと同様に，Prolog における計算とは，与えられた問い合わせに対して，関係性の記述から推論をすることで，該当する回答を求めることです．ここでは，以下の問い合わせを考えます．

```
# X と Y はProlog 変数です. この2行で親子関係をすべて出力します.
for sons in prolog.query("child(X , Y)"):
   print(sons["X"] , "is child of" , sons["Y"])
```

prolog.query メソッドを呼び出すことで，問い合わせを実行することができます．ここでは，「X はY の子供である」という問い合わせにより，X とY に当てはまるものを求めています．つまり，「親子関係にあるのはだれとだれですか」という質問をしているわけです．答えの出力が for 文になっているのは，解答は複数ある場合にすべてを出力するためです．この例では 1 つしかないので，以下のような出力が得られます．

```
jane is child of michael
```

親族関係の探索

続けて，少し複雑な親族関係を扱ってみましょう．

```
prolog = Prolog()

# Albertの子供たち
prolog.assertz("father(albert, bob)")
prolog.assertz("father(albert, betsy)")
prolog.assertz("father(albert, bill)")

# Aliceの子供たち
prolog.assertz("mother(alice, bob)")
prolog.assertz("mother(alice, betsy)")
prolog.assertz("mother(alice, bill)")

# Bobの子供たち
prolog.assertz("father(bob, carl)")
prolog.assertz("father(bob, charlie)")
```

上のプログラムは，Albert，Alice，Bob の子供たちを親子関係で記述しています．よく見ると，Albert と Alice の子供に Bob がいるので，Bob の子供 Carl と Charlie は Albert と Alice の孫であることがわかります．今度は新しい述語 mother が加わっていること，親子・祖父母と孫の関係を記述する述語がないことから，それらの述語に関する記述を追加します．

```
prolog.assertz("parent(X,Y) :- father(X,Y); mother(X,Y)")
prolog.assertz("grandparent(X,Y) :- parent(X,Z),parent(Z,Y)")
prolog.assertz("grandchild(X,Y) :- parent(Z,X) , parent(Y,Z)")
```

一番上では，親であることを表す parent という述語を導入しました．「X が Y の父親であるか，または，X が Y の母親であるならば，X は Y の親である」という意味です．Prolog ではセミコロン ; は論理和（「または」）です．一方，カンマ , で区切った場合は論理積（「かつ」）を表します．2 行目は「X が Z の親であり，かつ，Z が Y の親ならば，X は Y の祖父母である」ということです．なお，日本語で「祖父または祖母」のわかりやすい言い方が見当たらないため，ここでの「祖父母」は性別を気にしないことにしています．3 行目も同様に，「Z が X の親であり，かつ，Y が Z の親ならば，X は Y の孫である」という意味を表します． これで親子，祖父母と孫の関係が記述できました．

次に，問い合わせをしてみます．Albert が誰の祖父母か問い合わせてみましょう．変数 X を使って，問い合わせを下のように書くことができます．この場合も，X に入る答えが複数あるかもしれないので，for 文を使って，解答をすべて出力します．

```
for x in prolog.query("grandparent(albert , X)"):
    print("Albert" , "is grandparent of" , x["X"])
```

この問い合わせを実行すると，以下の解答が得られます．

```
Albert is grandparent of carl
Albert is grandparent of charlie
```

最後に，Carl が誰の孫かも問い合わせてみましょう．上と同様に問い合わせを書けばよいです．

```
for x in prolog.query("grandchild(carl , X)"):
    print("Carl" , "is grandchild of" , x["X"])
```

実行結果は，以下の通りです．

```
Carl is grandparent of albert
Carl is grandparent of alice
```

おわりに

今回の記事では，簡単な親族関係を例に使いながら，Google Colab が提供する Python の環境上で SWI-Prolog を実行する方法を紹介しました．今回紹介した方法については，一点だけ注意点があります．Google Colab で実行した場合は，Prolog の処理系で実行エラーが生じると，Google Colab の実行中のランタイムセッションがクラッシュして切断してしまうことが何度かありました．その場合は，セッションに接続して，最初からやり直してください（今回紹介した方法では，SWI-Prolog のインストールからやり直しになってしまいますが・・）．

Prolog は高機能なプログラミング言語で，リスト構造や算術演算を扱うこともできますし，Vol.165 の pyDatalog で取り上げた階乗のような再帰的な計算も実行することができます．Prolog は計算手順を記述しないため，基本的にどのように計算が進むかは Prolog の処理系に任せることになりますが，カットという仕組みを使うことで，計算を制御することもできます．また，今回の例では実行の途中経過を示すことはしませんでしたが，pyswip には Prolog から Python の関数を呼び出す機能があり，ログの出力のようなことも可能です．「はじめに」で挙げた例題の引用元では，もっと複雑なパズル問題も扱っているので，興味のある読者は見てください．

GSLetterNeo Vol.197
2025 年 3 月 20 日発行
発行者　株式会社 SRA 技術本部 先端技術研究室

編集者　熊澤努　方 学芬
バックナンバー https://www.sra.co.jp/public/sra/gsletter/
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8

夢を。Yawaraka Innovation
やわらかいのべーしょん