

鉄道路線モデルに基づくプログラムの動作の可視化 Visualization of Programs Based on Railroad Route Model

西尾 嘉矩[†]
NISHIO Yoshinori

六沢 一昭[‡]
ROKUSAWA Kazuaki

1 はじめに

本稿では、プログラムの動作を鉄道路線モデルに基づいて可視化するシステムについて述べる。

鉄道路線には、ループ状の路線や乗換駅がある。これらを見ると、プログラムにおける繰返し及び分岐に類似していることに気がつく。従って、プログラムを鉄道路線に置き換えることができれば、プログラムの動作を電車の動きによって可視化できる。

2 鉄道路線モデル

2.1 鉄道路線モデルとは

鉄道路線モデルとは、プログラムを鉄道路線に対応させたモデルである。具体的な対応を以下に示す。

- プログラムの制御構造（繰返し、分岐、call/return）を鉄道の路線図に対応させる。
- プログラムの実行の流れを、線路上を走る電車の動きに対応させる。

2.2 実際の鉄道路線における電車及び乗客の移動

実際の鉄道路線には、ループ状の路線や乗換駅がある。ループ状の路線では、電車はその路線を回り続ける。乗換駅では、乗客は次に乗る路線の電車へ乗り換える。乗換駅には、改札内で乗り換える乗換駅と、改札外で乗り換える乗換駅がある。

2.3 プログラムの制御構造と鉄道路線の対応

表 1: プログラムの制御構造と鉄道路線の対応

制御構造	鉄道路線
繰返し	ループ状の路線
分岐	乗換駅
call/return	乗換駅

- 繰返しは処理を繰り返す。これは、ループ状の路線を電車が回り続ける状況に類似している。
- 分岐は条件に応じて次の処理を選択する。これは、乗換駅で、改札内で乗り換える状況に類似している。
- call/return は、サブルーチン呼び出す/呼び出し側に戻る処理を行う。これらは、乗換駅で、改札外で乗り換える状況に類似している。

2.4 複数プロセスと電車の対応

ここまで1つの電車の動きに着目してきたが、実際の鉄道では複数の電車が走っている。この複数の電車はプログラムとどのように対応するだろうか。

1つのプロセスは1つの電車に対応するであろう。従って、複数のプロセスは複数の電車に対応する。そして、新たなプロセスの生成は新たな電車の生成に、複数のプロセスの実行は複数の電車の動きに対応する。

[†]千葉工業大学 大学院 情報科学研究科 情報工学専攻
[‡]千葉工業大学 情報科学部 情報工学科

3 可視化

本節では、鉄道路線モデルを用いたプログラムの動作の可視化[§]を述べる。

3.1 繰返し、分岐の可視化

繰返しはループ状の路線で表現する(図1(a))。繰返し実行中、電車はループ状の路線を矢印の向きに走る。分岐は乗換駅で表現する(図1(b))。電車は、乗客が乗り換えた先の路線を走る。

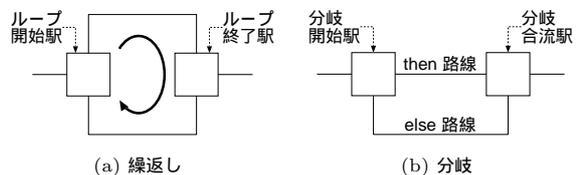


図 1: 繰返し、分岐に対応する路線図

3.2 call/return の可視化

call/return はともに乗換駅で表現する(図2)。call の可視化

1. 電車が call 駅に到着する。
2. サブルーチン開始駅から電車が出発する。

return の可視化

1. 電車が return 駅に到着する。
2. call 駅に停車していた電車が出発する。

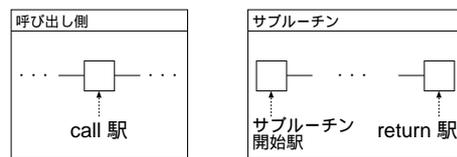


図 2: call/return に対応する路線図

3.3 fork (プロセスの生成) の可視化

fork は fork 駅で表現する。fork の実行は、以下のように fork 駅での新たな電車の生成で可視化する(図3)。

1. 電車 A が fork 駅に到着する。
2. fork 駅で電車 B が生まれ、電車 A は駅を出発する。
3. 電車 B が fork 駅を出発する。

4 表示範囲の指定

プログラムサイズが大きい時、プログラム全体ではなく、一部分のみを調べたいことがある。そこで、表示範囲を指定する機能をシステムに持たせた。

4.1 表示範囲の指定方法

プログラムソースに指示コメントを記述する、あるいは可視化画面において駅を指定することにより、表示範囲を指定することができる。

[§]本システムの可視化対象は制御構造のみである。

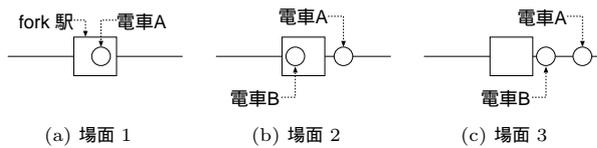


図 3: fork の可視化

単一指定 指示コメントの直後の制御構造のみを表示/非表示にする。#show と #hide がある。

領域指定 2つの指示コメントの間の全ての制御構造を表示/非表示にする。#begin show/#end show と #begin hide/#end hide がある。

4.2 表示の考察

制御構造はネストすることがある。ネスト内の制御構造にのみ表示/非表示指定をした場合、その親や子の制御構造はどう扱うべきだろうか。

ネストした制御構造は木構造で表現できる(図4)。節Bのみ非表示にすると、節Bの親である節Aと、節Bの子である節C/Dの関係が分断される。従って、ある節を非表示にする場合、その子は非表示にする。また、根は表示するのが適当であろう。そのため、節Bのみ表示すると、やはり節Bと、その親である節Aとの関係が分断されてしまう。従って、ある節を表示する場合、その親は表示する。

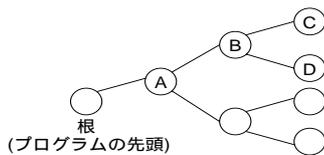


図 4: 木構造

5 実行例

図5(a)のプログラム*の実行の可視化の一場面を図5(b)に示す。図5(b)の黒丸は電車を示す。

図5(a)のプログラム

3行目に#begin show, 17行目に#end showがあるため、4行目~16行目の全ての制御構造を表示する。しかし、10行目に#hideがあるため、11行目のfor文と、その子にあたる12行目のif文は非表示にする。

図5(b)の電車の動き

入力「0」の時**の電車の動きを以下に示す。

1. 「START 駅」を出発し、「駅 1(分岐開始駅)」に到着する。
2. then 路線を走り、「駅 2(ループ開始駅)」に到着する。
3. ループ路線を3周し、「駅 3(ループ終了駅)」に到着する。
4. 「駅 3」を出発し、「駅 4(分岐合流駅)」を通過して「GOAL 駅」に到着する。

可視化画面での表示範囲の指定

図5(b)の「駅2」に、非表示の単一指定を行った結果の路線図での可視化の一場面を図6に示す。

図6の電車の動き

1. 「START 駅」を出発し、「駅 1」に到着する。
2. then 路線を走り、「駅 4」を通過して「GOAL 駅」に到着する。

*現在, Perl プログラムのみが可視化対象である。

**変数\$flag に0が代入される。

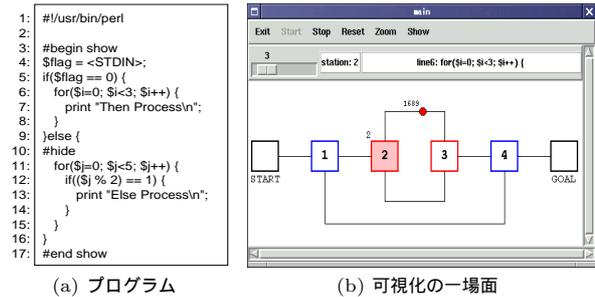


図 5: 可視化例

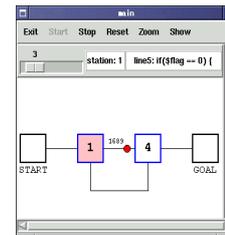


図 6: 指定結果

6 評価

Perl プログラミングの経験がある情報科学部の学生11人に本システムを使用してもらった。以下は寄せられたコメントの一部である。

- プログラムと鉄道路線の対応関係が分かりやすい。
 - プログラムのバグを見つけることができた。
 - 調べたい部分だけ表示することができ、路線図が分かりやすくなった。
 - 表示範囲の指定前と指定後で、路線図のどこが変化したかが一目で分かる表示がほしい。
- 上記のコメントより、以下のことが考えられる。
- プログラムを鉄道路線に対応させた鉄道路線モデルは有益である。
 - デバッグに有益である。(電車の動きによって、予想外の実行部分を見つける, など)
 - 調べたい部分だけ表示できることは有益であるが、さらなる改良が必要である。

7 まとめ

鉄道路線モデルを説明し、このモデルに基づいてプログラムの動作を可視化するシステムについて述べた。本システムを使用すると、ユーザは、プログラムの制御構造(繰返し, 分岐, call/return)を鉄道の路線図で、プログラムの実行の流れを電車の動きで見ることができる。また、新たなプロセスの生成を新たな電車の生成で、複数のプロセスの実行を複数の電車の動きで見ることができる。さらに、表示範囲を指定することで、特定の制御構造のみ表示することができる。

参考文献

- [1] 西尾 嘉矩, 六沢 一昭: 鉄道路線モデルに基づくプログラムの動作の可視化, FIT2006 情報科学技術フォーラム講演論文集, B-024, pp.123-124, 2006.