GSLetterNeo vol.136

2019 年 11 月

形式手法コトハジメ

–TLA$^+$ Toolbox を使って (4)-

熊澤 努 kumazawa @ sra.co.jp

はじめに

GSLetterNeo Vol.134 までで扱っていた仕様は非常に簡単なもので、繰り返しや分岐などの制御機構を全く使っていませんでした。より複雑な仕様を記述できるようになるには、PlusCal が用意している構文を活用していく必要があります。今回は、信号機仕様を拡張しながら、繰り返し構造、条件分岐、非決定的な分岐を使った少し複雑な仕様を書いていきましょう。

繰り返しを書く

Vol.134 まで見てきた信号機の仕様には大きな問題点があります。それは、点灯回数です。普通の信号機は赤→青→黄を繰り返すのに、仕様に書いた信号機は、青と黄色は 1 度だけ、赤は初期状態を含めて 2 度しか点灯しません。そこで、赤、青、黄の点灯を繰り返すように仕様を変更することにしましょう。次のページに、PlusCal で書いた変更後の仕様を示します。

繰り返しは、PlusCalの`while`構文を使って書くことができます。プログラミング言語の使用経験のある読者には馴染みのある概念でしょう。繰り返しの記述はラベル`Signal`の次の

```
------------------------------- MODULE spec -------------------------------
(* --algorithm TrafficLight
variables
  light = "red";
begin
  Signal:
    while TRUE do
      Green:
        light := "green";
      Yellow:
        light := "yellow";
      Red:
        light := "red";
    end while;
end algorithm;*)
=========================================
```

行のwhileからです。繰り返しは、繰り返しを継続するか停止するかの判定を書いた「条件」と繰り返す内容を書いた「本体」から成ります。条件はwhileの直後に書きます。条件の後にはdoを書き、本体はその後からend while;までの間に書きます。なお、本体にラベルがある場合にはwhile構文全体にもラベルが必要です。そのため、今回の仕様にはラベルSignalが追加されています。これは繰り返しの先頭を指し示しています。

条件は繰り返しのたびに本体に先立って評価されます。条件が成り立つ、つまり真(PlusCalではTRUE[1])と評価されると本体が続けて評価され、その後再び条件が評価されます。一方、条件が不成立、つまり偽(PlusCalではFALSE)と評価された場合には、本体は無視されて終了となります。

この仕様の条件と本体について詳しく見てみましょう。ここでは TRUE が条件です。したがって、この条件は常に成立するので、無限回の繰り返しであることを意味しています。条件には、例えば x < 1 のように、TRUE または FALSE と評価される式を書いても構いません。

本体には、以前の信号機の仕様と同じ青→黄→赤への灯火の遷移が書かれています。

以上をまとめると、今回拡張した仕様は、信号機の灯色の変化の系列「赤→青→黄→赤→青→黄→赤→…」を表していることになります。無限回の繰り返しは、信号機が通常は停止することなく動作し続ける機器であることを書きあらわしたものと考えると、理解しやすいでしょう。

[1] TRUE や FALSE は PlusCal で予約された値で、真理値と呼ばれています。プログラミング言語 C++の bool 型や Java の boolean 型、Python の Bool 型の値に相当します。

この仕様が構文上正しい記述であることも確認しておきましょう。Vol. 134 の記事のとおり、**File** メニューから **Translate PlusCal Algorithm** を選択すると、対応する TLA⁺の仕様が出力されるので試してみてください。

条件分岐を書く

条件に応じた分岐処理も、PlusCalのif構文を使うと書くことができます。信号機の例を使って、条件分岐の書き方を見ることにしましょう。
ここで考える信号機は、黄色が点灯するかどうかをあらかじめ選べる仕様であるとします。それを盛り込むために、新たに変数use_yellowを宣言して、その値TRUE/FALSEに応じて黄色の灯火機能を含めるか否かを分岐するようにします。また、簡単のため、use_yellowの初期値はTRUEとします。if構文による条件分岐を使った仕様は下のようになります。

```
------------------------------ MODULE spec ------------------------------
(* --algorithm TrafficLight
variables
  light = "red", use_yellow = TRUE;
begin
  Signal:
    if use_yellow then
      Green:
        light := "green";
      Yellow:
        light := "yellow";
    else
      light := "green";
    end if;
end algorithm;*)
=============================================
```

条件分岐はif ... then ... else ... end if;の形式で書きます。ifの直後にTRUE/FALSEと評価される条件を書きます。上の仕様では変数use_yellowの値が条件です。thenの後には、条件がTRUEの場合に続けて評価される内容を書きます。ここでは、青と黄が点灯する信号機の挙動を書いています。elseとend if;の間には条件がFALSEの場合に評価される内容を書きます。上の例で、青への点灯のみが記載されている箇所がそれにあたります。条件が一つではなく、例えば青のみを点灯する条件がある場合には、else分岐をelsif ... thenとすることで第二の条件を書くことができます。また、else分岐は無くても構いません。なお、ラベルのつけ方は繰り返しの場合と同様で、if構文内でラベルGreenとYellowが記されているため、if構文の先頭にラベルSignalが必要です。

非決定的な振る舞いを書く

PlusCal には、挙動の選択を記述するために、条件分岐とは異なる方法が用意されています。例として、最初に赤が点灯している信号機について、その次に点灯するのは青か黄のどちらかである場合を考えましょう（このような信号機はないと思いますが、ここでは目を瞑ってください）。このような、青か黄かどちらか事前には決まらない振る舞いを指して非決定性といいます。聞きなれない言葉かもしれませんが、TLA$^+$ Toolbox では、非決定性は「青の点灯と黄の点灯どちらも起こりうる分岐」と考えていただくと少しわかりやすいかもしれません。

非決定的な振る舞いをする信号機の仕様は下のようになります。

```
------------------------------- MODULE spec -------------------------------
(* --algorithm TrafficLight
variables
  light = "red";
begin
  either
    light := "green";
  or
    light := "yellow";
  end either;
end algorithm;*)
=============================================
```

信号が青か黄のどちらかに変わるという挙動は、`either` 構文を使って `either ... or ... end either;`とすることで書くことができます。ここでは分岐は 2 通りですが、`or` 以下を追加すると分岐の数を増やすことができます。この構文は、非常に複雑な振る舞いや並行動作を簡潔に書くのに適しています。ここでの例で考えると、信号の色の変化が、例えば時刻や設置場所など信号機以外の外界の要因で決まる場合を、簡単に表現することができます。したがって、ユーザの入力にしたがって応答するシステムや、仕様の作成者が詳細な振る舞いを知ることができないブラックボックス型システムの記述に有効です。なお、非決定的な振る舞いは `with` 構文を使って書くこともできますが、本稿では省略します。

おわりに

本稿では、PlusCal における制御構造の書き方を説明しました。繰り返しや条件分岐は、プログラミングを経験したことのある読者にはそれほど難しくはないかと思います。一方、非決定性はプログラムを書く際に意識することはあまりないかもしれません。どちらもよく使われますので、いろいろな仕様を眺めながら慣れていただくのがよいでしょう。

次回は、PlusCal で使用可能なデータ構造について解説したいと思います。これまで特に断らずに、文字列や真理値(`TRUE`/`FALSE`)を使ってきました。PlusCal では、これらに加えて数値やより複雑な構造をもったデータやそれらの演算を扱うことができます。データについて分かれば、いろいろな種類の仕様を書くことができるので、ぜひ身につけていただきたいと思います。

GSLetterNeo Vol.136
2019 年 11 月 20 日発行
発行者　株式会社 SRA 先端技術研究所

編集者　土屋正人
バックナンバー http://www.sra.co.jp/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