GSLetterNeo vol.132

2019 年 7 月

形式手法コトハジメ

–TLA$^+$ Toolbox を使って (2)-

熊澤 努 kumazawa @ sra.co.jp

はじめに

GSLetterNeo Vol.130 で TLA$^+$ Toolbox を紹介しました。今回からより詳しく TLA$^+$ Toolbox について説明していきます。今回は、TLA$^+$ Toolbox のインストールの仕方から、簡単なモデルを書くところまで解説したいと思います。なお、本稿で使用する TLA$^+$ Toolbox は、執筆時点[1]で最新のバージョン 1.5.7 です。

インストール

TLA$^+$ Toolbox は Java 環境上で動作するツールです。TLA$^+$ Toolbox を PC にインストールする前に、まず Java をインストールしてください。Java のインストールの仕方については省略します。

TLA$^+$ Toolbox は、公式サイト[2]から次の手順でダウンロードします。

[1]本稿は 2019 年 6 月現在の情報をもとに執筆しています。最新の情報については公式サイトを参照してください。

[2] http://lamport.azurewebsites.net/tla/tla.html

1. 公式サイトのリンク **The Toolbox** をクリックして、TLA+ Toolbox の解説サイト[3]に移動します。
2. 解説サイトのメニュー **Obtaining the Toolbox** をクリックします。
3. 子メニュー **Downloading the Toolbox** をクリックします。**-- DOWNLOAD FROM GITHUB --** と **-- ALTERNATE DOWNLOAD SITE --** という二つのリンクが表示されることを確認してください。
4. [GitHub からダウンロードする場合] **-- DOWNLOAD FROM GITHUB --** をクリックして、GitHub に移動します。インストールする環境 (Linux, MacOSX, Win32) に合わせて、*TLAToolbox-1.5.7-[環境名].zip* をクリックすれば、ダウンロードが始まります。なお、ソースコードもダウンロードできるので、このツールの技術的な詳細に興味のある方は一緒にダウンロードしましょう。
5. [専用サイトからダウンロードする場合] **-- ALTERNATE DOWNLOAD SITE --** をクリックして、ダウンロードサイトに移動します。ステップ 4 と同様に環境に合致する zip ファイルを選択すれば、ダウンロードが始まります。

インストールは非常に簡単で、ダウンロードした zip ファイルを解凍すれば完了です。システム設定の変更などの作業は不要です。

TLA$^+$ Toolbox を使う上で便利なツールも、必要に応じてインストールしましょう。TLA$^+$ Toolbox には、作成した仕様を pdf で出力する機能や、検証結果を図式表現に変換する機能がありますが、これらの機能は TLA$^+$ Toolbox には組込まれていないツールによって実現されています。そのため、使用するためには、あらかじめ必要な無料ツールをインストールしておく必要があります。仕様の pdf 出力機能を使うには、LaTex あるいは Tex をインストールしてください[4]。また、図式化機能を有効にするには、Graphviz[5]をインストールします。これらのツールは必須ではないので、インストールしなくても TLA$^+$ Toolbox の主要な機能を使うことができます。後で必要になった際にインストールしても構いません。

---

[3] https://lamport.azurewebsites.net/tla/toolbox.html

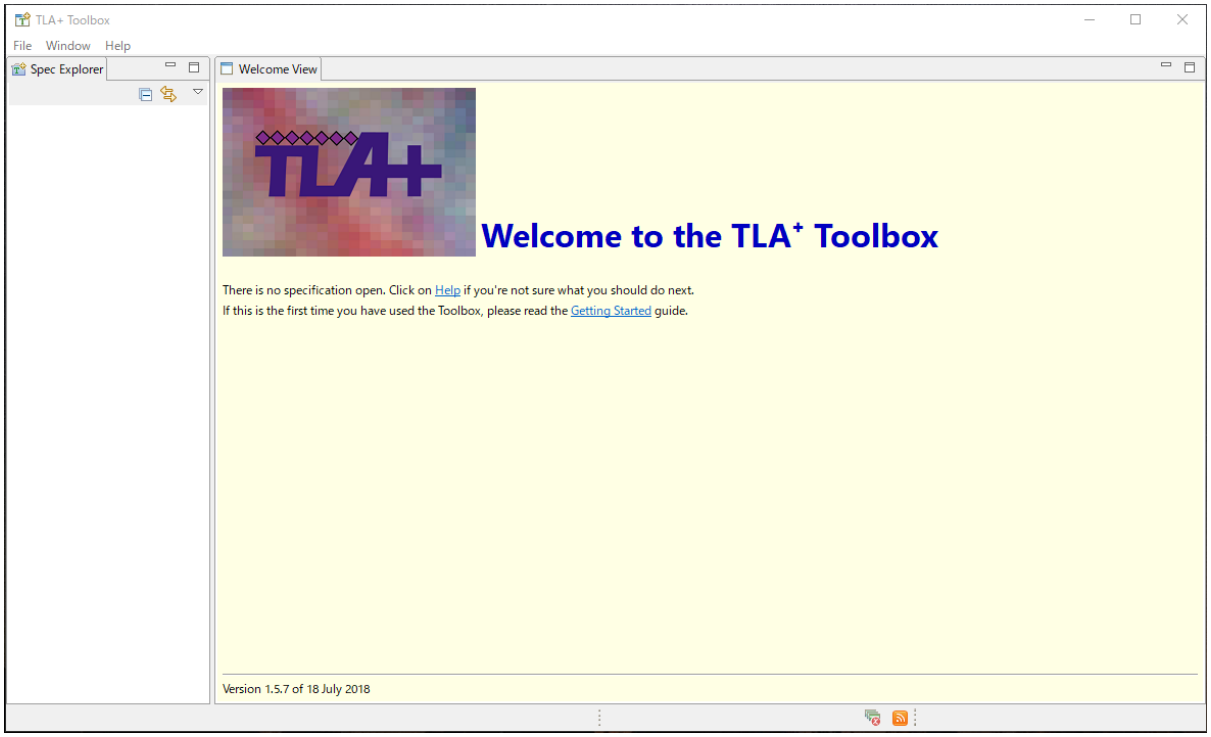
[4] LaTex のインストール方法は使用環境や配布されているツールによって異なるので、注意が必要です。詳しくは、例えば、LaTex Project (https://www.latex-project.org/) 、The TeX Users Group (https://www.tug.org/) 、日本語 TeX 開発コミュニティ(https://texjp.org/) などのサイトを参照してください。

[5] Graphviz についての詳細は、公式サイト (https://graphviz.gitlab.io/) を見てください。

アンインストールする場合には、解凍した全てのファイルを手作業で削除してください。

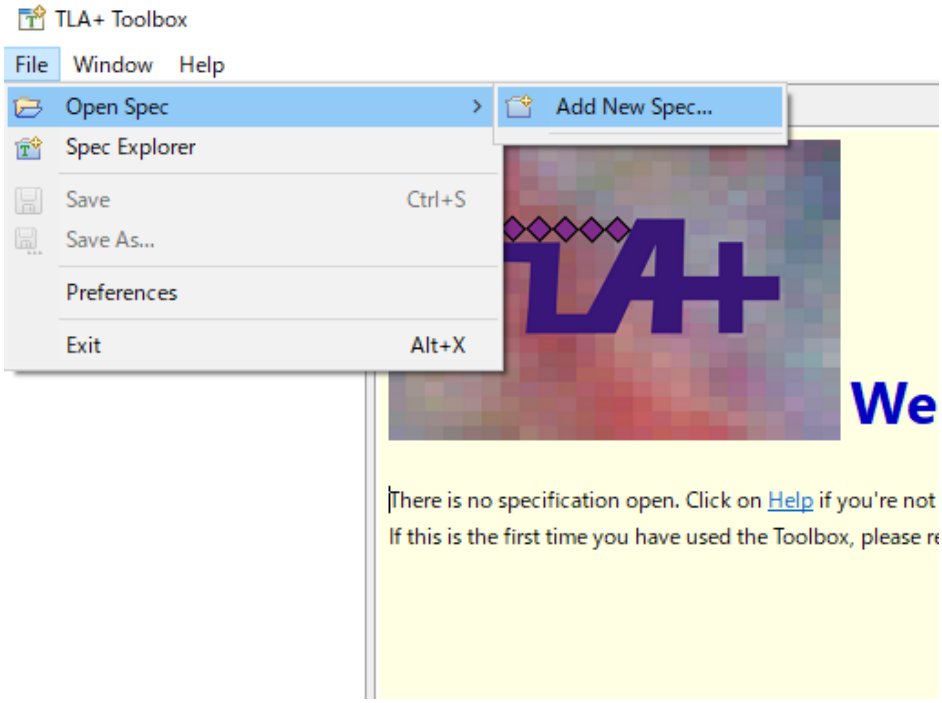
TLA+ Toolbox を起動する

インストールを終えたら、TLA+ Toolbox を起動しましょう。起動の仕方は使用環境により多少異なります。筆者が使用している Windows の場合には、解凍してできたディレクトリにある toolbox ディレクトリ内の toolbox.exe が TLA+ Toolbox の実行ファイルです。早速実行してみましょう。ロゴが表示された後に、下の図のような画面が表示されれば成功です。

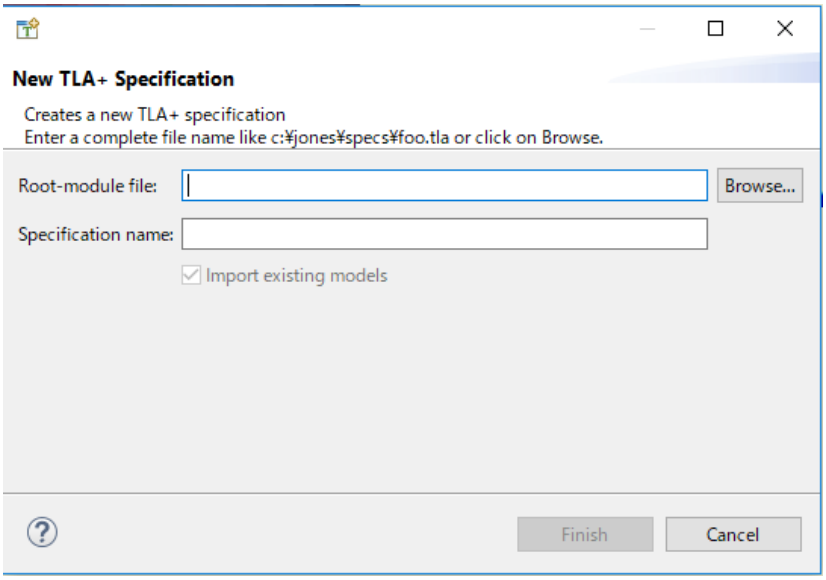


仕様を記述するファイルを作る

仕様を記述するためには、専用のファイルを作成する必要があります。次の図のように、**File** メニューから **Open Spec** → **Add New Spec...** と選択してください。

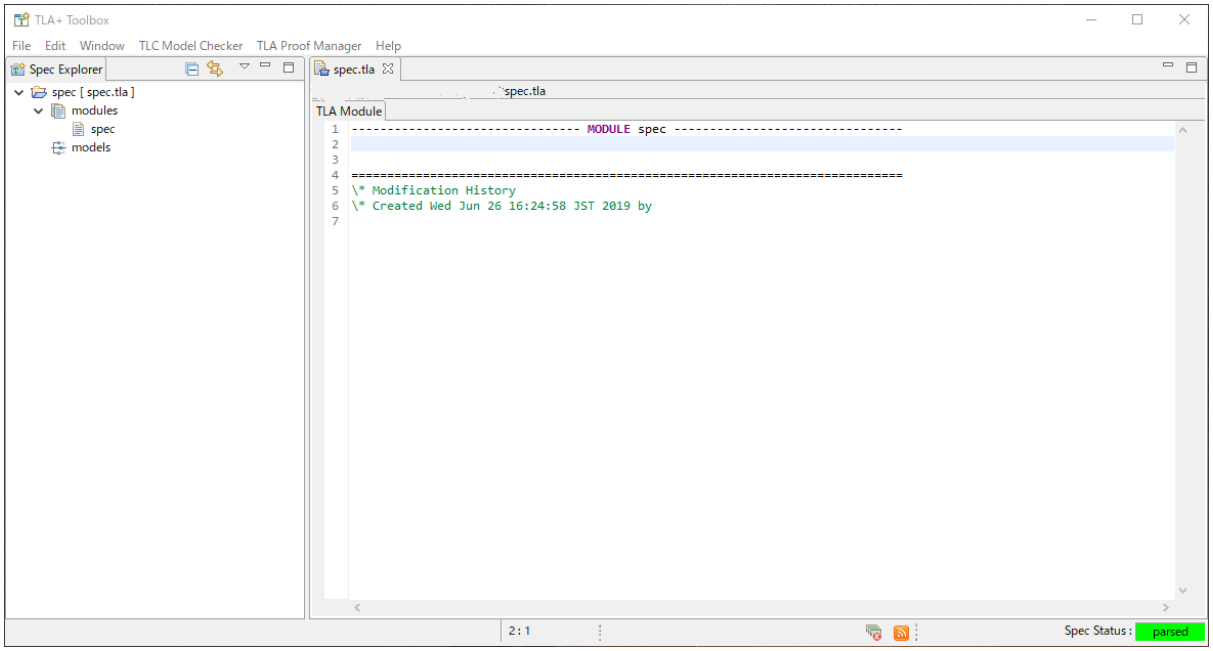


すると、下図のようなダイアログが表示されます。ダイアログの **Root-module file** 欄には、仕様を記述するファイルのパスを入力します。ファイルの拡張子は.tla とします(以下、tla ファイルと呼びます)。tla ファイルを入力すると、**Specification name** 欄に拡張子を除いたファイル名が自動的に入力されます。Specification name は変更しても構いません。次に、**Finish** ボタンを押すと、tla ファイルが自動的に作成されます。既存の tla ファイルを指定して開く場合も同じ手順を実行してください。



次の図は、spec.tla を新規作成した後の画面表示です。画面左の **Spec Explorer** に、作成した仕様の情報が表示されています。tla ファイルは modules という階層の下に追加され

ます。models という階層もつくられていますが、こちらは仕様の検証の際に使いますので今は空です。画面右の spec.tla というタブに spec.tla の内容が表示されています。



簡単な仕様を書く

spec.tla を編集して仕様を書いてみましょう。

準備として、作成されたファイルの内容を詳しくみておきます。spec.tla ファイルの主要部分を以下に示します。最初の行には、tla ファイルのモジュール(MODULE)名 spec が記述されています。ファイルのモジュール名はファイル名と同一でなくてはいけません。変更するとエラーになります。また、モジュール名の前後のハイフン(-)列は削除しないでください。これは TLA$^+$の予約語で、仕様記述の先頭を表す識別子です。同様に、最後の行のイコール(=)列も仕様の終端を示す予約語ですので、削除してはいけません。仕様は 2 行目からイコール行の直前の行に記述します。*は単一行コメントの開始を意味します。改行までの文字列が無視されます。

```
---------------------------------- MODULE spec ----------------------------------
\* ここに仕様を記述する。ここ以外の場所に書いても無視されるので注意。
================================================================================
```

以上の準備の下で、TLA+ Toolbox がサポートする仕様記述言語のうち、PlusCal で実際に仕様を書いてみます。以下に、信号機の挙動を単純化して記述した仕様を示します。

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

PlusCal の記述は--algorithm から end algorithm;までです。先頭の 2 つのダッシュ(--)や最後のセミコロン(;)も必要ですので、削除してはいけません。文字列 TrafficLight は仕様の名称です。モジュール名と一致している必要はありません。適切な名称を付けましょう。

PlusCalでは、最初に仕様で使うすべての変数をvariables以下に宣言します。ここでは、灯色を表す変数lightを宣言しています。さらに、この変数は、文字列"red"を初期値とすることが記述されています。変数lightを宣言した行の字下げは、必須ではありませんが、適当にしておくと仕様が見やすくなります。ただし、TLA+ Toolboxではタブ文字が使えないので注意しましょう。変数宣言はセミコロンで終えます。

次に、begin以下に記述対象の挙動を仕様として記述します。今回の例では、信号機の灯色の変化「赤→青→黄→赤」を記述しています。変数lightで現在の灯色情報を管理することにします。

文light := "green";は変数lightに文字列"green"を代入することを意味します。これは、信号の色が青になったことを表現しています。ここで、変数宣言での初期化と異なり、代入をコロンとイコール(:=)で書くことに注意してください。セミコロンは文末を表します。

コロンで区切られたGreenは、文につけるラベルです。ラベルは一つまたは複数の文をまとめて単一処理とすることを意味し、並行システムの記述の際に重要な働きをします。信号機の例では各文にラベルを付けていますが、必ずしもすべての文につける必要はありません。ラベルの記述直後の文から新しいラベルの直前までの文が、そのラベルの有効範囲となります。一つのラベルの有効範囲内では、各変数への代入が一度しか許されていないため、慣れるまでは各文につけておくのがよいと思います。

最後に、仕様全体を(**)で括ります。これは PlusCal の予約語ではなく、もう一つの仕様記述言語 TLA$^+$のブロックコメントです。実は、TLA$^+$ Toolbox は TLA$^+$で書かれた仕様を検証などの際に扱いますが、PlusCal を直接取り扱うことはできません。PlusCal は TLA$^+$のコメントとして記述します。

おわりに

本稿では、TLA$^+$ Toolbox のインストール、起動方法、記述言語 PlusCal を用いた仕様記述の基本を解説しました。今後も引き続き、PlusCal を主に使いながら、仕様を形式的に書いたり、検証したりする方法を解説していきたいと思います。

GSLetterNeo Vol.132
2019 年 7 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