5L-06

UMLモデリングツールによる分散アルゴリズムの記述とモデル検査器SPINとの連携

Description of Distributed Algorithms using A UML Modeling Tool and Linkage to The Model Checker SPIN

萬田 悠 †　　**和崎 克己** ††
Yu Manta　Katsumi Wasaki

1　はじめに

モデル検査は検査の対象となる仕様の振る舞いにおいて，モデルが初期状態から取りうる状態を自動的に網羅し，調べる技術である．そこで，モデル検査ツール SPIN を使用して検査する対象のモデルを UML 図で記述する．UML（Unified Modeling Language）はソフトウェア開発の上位設計において，データの構造や処理の流れなどを図示するための記法を定めたものである．本研究では提案手法として，分散アルゴリズムを対象とし，SPIN の記述言語である PROMELA の通信チャネル記述方法に変更を加え，合成構造図とステートマシン図を使用して記述する．さらに，作成した UML 図から自動で PROMELA のコードを生成するための前段階として，UML モデリングツールである astah* professional [1] と SPIN の反例可視化ツールである ispin を連携し，SPIN モデル検査器で取得した反例ファイルの文字列を astah* 上に反映させるフレームワークの作成を行う．

2　UML を用いた PROMELA の記述

UML 図を用いた PROMELA の記述にあたっては，合成構造図でプロセスのインスタンスとチャネルを定義し，ステートマシン図でプロセスの振る舞いを定義する．その際，ユーザは以下で記述するガイドラインに従って，対象となるモデルを記述する [2].

2.1　インスタンスの定義

モデル記述言語 PROMELA は，プロセスの数だけ振る舞いを記述する．本研究でのプロセスの数は，合成構造図で定義した構造化クラスの数とする．astah* のハイパーリンクの機能を用いて，構造化クラスとステートマシン図を対応させる．ハイパーリンクはファイル，URL，プロジェクト内の図要素，モデルのいずれかを構造化クラスに設定し，astah*上でリンク付けすることができる機能である．

2.2　変数と型の定義

astah* では様々な要素に名前と，その値を紐づけることが可能なタグ付き値が存在する．そこで，本研究ではモデルが持つ固有の ID とその値の型を各構造化クラスにタグ付き値 [value] と [type] として定義した．また，astah* の UML 図には自由にコメントを記述できるノートが存在し，上記以外の各プロセスが持つ変数とその型はステートマシン図のノートで定義し，全てのプロセスで扱うグローバル変数とその型は合成構造図のノートで定義した．

2.3　通信チャネルの定義

本研究でのツール連携に関する提案手法では 1，従来の通信チャネルの記述方法では困難さが生じるため，以下のような変更を行った．従来の PROMELA での通信チャネルの定義と送信は以下のように記述される．

```
1 chan line[3] = [2] of {int};
2 line[0]!m;
```

上記の例では，配列で宣言されている line というチャネル名の 0 番目を用いて m というメッセージの送信を行うことを表している．提案手法では新たに，各プロセスが送信動作で使用するポートを以下のように define 文を使用して，マクロで定義する．また，受信も送信の記述と同様の変更を加えた．

```
1 chan line[3] = [2] of {int}
2 #define port_01 1;
3 #define send(port_ID,m) line[port_ID]!m;
4 send(port_01,value);
```

この手法では，どのプロセスがどのチャネルを使用するのかを各プロセスの内部で決定するのではなく，プロセスの外部のマクロの定義で決定しており，チャネルを合成構造図で記述しやすく，容易にチャネルの接続状況を変更できる．

2.4　assert 文の定義

PROMELA の assert 文とは，SPIN でモデルが満たすべき性質を記述する方法の一つで，モデル中の任意の位置に挿入することができる．本研究では assert 文をステートマシン図のトランジションにおけるガード条件に記述すると定める (図 1).

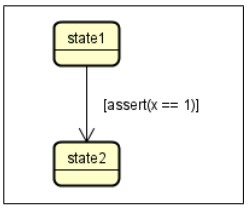

図 1　UML ステートマシン図における assert 文の記述

† 信州大学大学院総合理工学研究科, Graduate School of Science and Technology, Shinshu University
†† 信州大学工学部電子情報システム工学科, Department of Electrical and Computer Engineering, Faculty of Engineering, Shinshu University

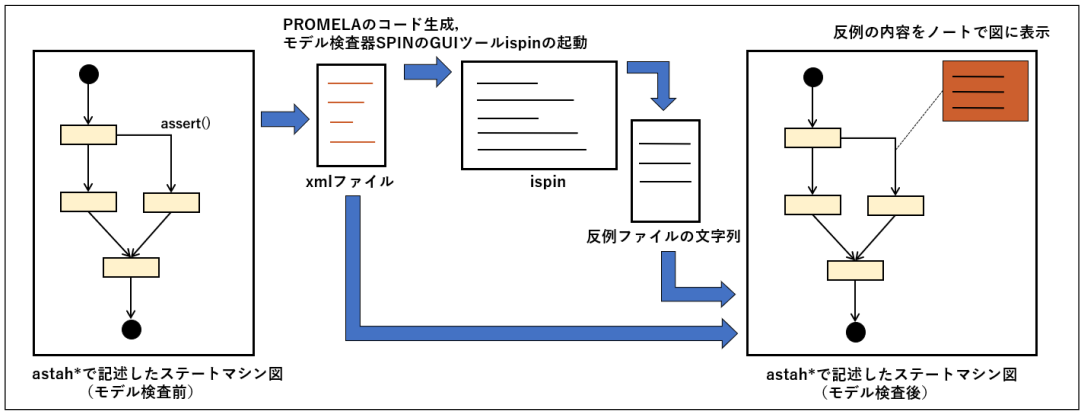

図 2　astah* と SPIN の連携イメージ図

3　モデル検査器 SPIN との連携

図 2 は本研究における，astah* とモデル検査器 SPIN との連携のイメージ図である．まず，astah* で記述した UML 図の情報を DOM を使用して保存し，XML ファイルを出力する．その後，XML ファイル情報を基に PROMELA のコード生成と SPIN の GUI ツールである ispin を起動し，SPIN によるモデル検査を実行する．最後に，取得した反例ファイルの文字列と XML ファイルの情報を照合し，反例箇所の特定と astah* の図に反映させる．

3.1　DOM を使用した UML 図情報の保存

DOM(Document Object Model) は XML 文書を取り扱うための API で，データをオブジェクトの木構造モデルで表現することができる．今回は合成構造図の構造化クラスを一番上の階層に位置づけ，その下に各構造化クラスにハイパーリンクされたステートマシン図の情報と，ステートマシン図が持つ状態やトランジションなどの要素の情報を追加する．また各ノードには，それぞれ astah* から取得した固有の ID やその要素が保持する情報を追加する．図 3 は XML ファイルの内容を一部抜粋したものである．ステートマシン図においては，状態の他にトランジションの情報も追加しており，UML 図の解析を行った際にトランジションに記述された assert 文を発見すると，モデル検査後にどの assert 文に問題が起きたのか特定するために自動的に固有の文字列が割り振られる．下記の図 3 の例では “assert” という属性に “TRUE001” という文字列が追加されている．

```
<StateMachine id="1qjv-4d9" name="sample">
<state id="1bq-a7e2_state1">
<transition assert="TRUE001" id="a70-a7e2"/>
</state>
```

図 3　DOM で保存した XML ファイルの内容

3.2　astah* professionl と ispin の連携

Windows 環境で ispin を使用するためには，いくつかのソフトウェアのインストールが必要である．astah* にはそれらのソフトウェアのインストールの確認と設定を行う新たな拡張タブを作成し，設定されたパスを使用して，ispin の起動を行うボタンを追加した．また，Mac OS においても同様の動作を行うように変更を加えた．

3.3　反例ファイルの出力

反例ファイルの内容を astah* に反映させるために，発生した assert がステートマシン図のどこに記述された assert 文か特定する必要がある．そこで，図 3 のように自動で各 assert 文に割り振った文字列をコード生成際に，コード中の assert 文に埋め込む．埋め込まれた文字列自体に意味は存在しないので，本来ユーザが検証したい事象に影響は与えない．現在，コード生成の機能は未実装なので，以下のようにコードが生成されると想定している．

```
1 #define TRUE001 1
2 #define TRUE002 2
3 assert((TRUE001)&(x==1));
```

```
spin: test.pml:3, Error: assertion violated
spin: text of failed assertion:
                          assert((1&(x==1)))
```

図 4　SPIN で取得できる反例ファイルの文字列

上記のコードから取得できるであろう反例ファイルの文字列は図 4 となり，この assert 文の直後の “1” という文字列と XML ファイルの文字列を照らし合わせることで，反例ファイルの文字列をノートとして出力する astah* の要素を特定する．

4　まとめと今後の課題

今回は作成した UML 図から PROMELA への自動コード生成の前段階として，astah* とモデル検査器 SPIN の連携を行った． ユーザはこの機能を使用することで astah* という一つのツールの中で UML 図の作成，モデル検査，反例ファイルの確認を行うことが可能となった．今後は未実装である自動で PROMELA のコードを生成する機能の追加，LTL 論理式や progress ラベルなどの assert 文以外のモデル検証への適応が必要である．

参考文献

[1] 株式会社チェンジビジョン：astah*professional, http://astah.change-vision.com/ja/product
[2] 萬田悠，和崎克己．“UML モデルの合成構造図とステートマシン図を用いた分散アルゴリズムの記述と整合性検査”，第 21 回情報科学技術フォーラム（FIT2022）講演論文集，Vol.1, No.A-001, pp.91-94, Sep. 2022.