B-010

PROMELA 自動コード生成を目的とした 複数の UML 図による分散アルゴリズムの記述方法

Method of Describe Distributed Algorithms Using Multiple UML Diagrams for Automatic PROMELA Code Generation

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．その後，それらの UML 図の情報を基に，動的型付けのプログラミング言語 Groovy [1] を用いて，テンプレートベースで PROMELA の自動コード生成を行う．

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，ユーザー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．

2.1　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，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は合成構造図のノートで定義した．

† 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

2.3　通信チャネルの定義

本研究では，従来の PROMELA の通信チャネルの記述方法では，UML 図で記述する際に，困難さが生じるため，以下のような変更を行った．従来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．

3　PROMELA の自動コード生成

図 1 は本研究における，astah* を用いた UML 図の作成から，PROMELA コードの自動生成と SPIN を使用したモデル検証までの流れを表したイメージ図である．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，Groovy のテンプレートエンジンを使用して，XML ファイル情報を基にテンプレートベースでの PROMELA の自動コード生成を行う．最後に，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し，その結果を astah* の図に反映させる．これまでの研究で，作成した UML 図の整合性検査と UML モデリングツールである astah* professional [2] と ispin の連携はプラグインで実装済みである [3, 4].

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

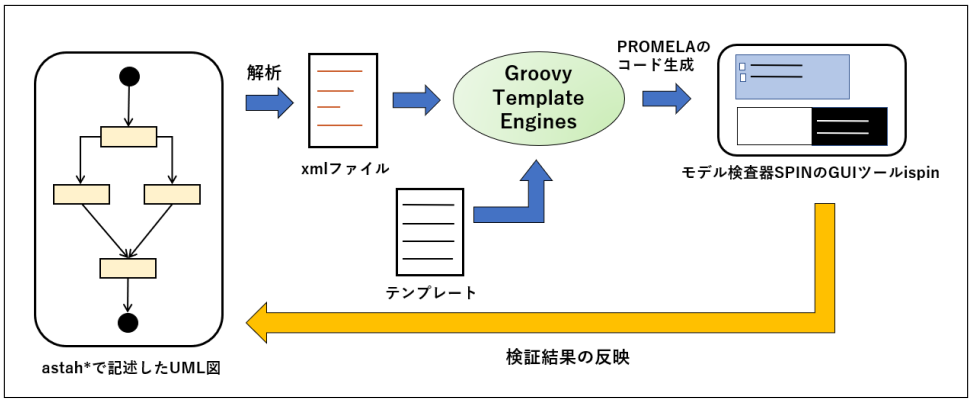

図 1　astah* と SPIN の連携イメージ図

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 1 は DOM で保存した 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，現在の状態から次の遷移先の状態を指す，“target” や分岐するための条件文を記述した “gurad” などが存在する．

ソースコード 1　XML ファイルの内容

```
<StateMachine id="1qjv-4d9" name="sample">
<state id="1qsu-4d9_state">
<transition id="a70-a7e2"
target="1sab-4d9_state" guard="x==1"/>
</state>
```

3.2　Groovy Template Engines

ソースコード 1 の XML ファイルと自身で作成したテンプレートを使用して，Groovy で PROMELA のコード生成を行う．Groovy は Java から派生した動的型付けのプログラミング言語で，今回はテンプレートから、動的に変数や条件分岐、ループを埋め込むことが可能な，テンプレートエンジンを利用する．

ソースコード 2　PROMELA テンプレート

```
#define send(port,m) msg_buffer[port]!m;
#define receive(port,m) msg_buffer[port]?m;
chan msg_buffer[122]=[BUF_LEN]of{<$type>};
<%for(port in port_List){%>\
#define<%= port_id.getAttribute("id")%>
<%= port_id.getAttribute("num")%>
<%}%>
<%for(procname in map.keySet()){%>
active proctype<%= procname %>(){
<%for(statename in map.get(procname)){ %>
<%=statename.getKey().getAttribute("id")%>:
<% for (trn in statename.getValue()) { %>
  ::<%= trn.getAttribute("trigger")%> ->
      goto<%= trn.getAttribute("target")%>;
<%}%>
```

上記のソースコード 2 は作成した PROMELA テンプレートを一部抜粋したものである．テンプレートには，Groovy で XML ファイルの情報を，map に格納してから渡しており，1 行目から 3 行目にかけて通信チャネルの定義を行い，4 行目から 7 行目でポートの定義を行っている．さらに，8 行目以降はプロセス名前や状態とトランジションの情報をループを使用して，map から順番に取り出している．

3.3　テンプレートベースのコード生成

下記のソースコード 3 は生成後の PROMELA コードの一部抜粋である．テンプレートに XML ファイルの情報を埋め込んで，テキスト形式で生成している．

ソースコード 3　生成後の PROMELA コード

```
#define send(port,m) msg_buffer[port]!m;
#define receive(port,m) msg_buffer[port]?m;
chan msg_buffer[122]=[BUF_LEN]of{<int>};
#define 7ou-4d92_P_in 0
#define 76k-4d92_P_out 0
active proctype 21-4d920_node_0 (){
  do
  ::receive(P_in,letter) ->
          goto ckh-a7e2_receive;
  od
```

4　まとめと今後の課題

今回は複数の UML 図を使用した，分散アルゴリズムの記述方法を定義した．また，それらの UML 図の情報を基に，Groovy を使用して，テンプレートベースで PROMELA コードの自動生成を行った．今後は，今回作成した Groovy を用いた自動コード生成の機能を astah* のプラグインへ実装する．また，意図的に不具合が存在する UML 図を作成し，コード生成から検証まで行うことで，プラグインの評価を行う必要がある．

参考文献

[1] Groovy：https://groovy-lang.org/
[2] 株式会社チェンジビジョン：astah*professional, http://astah.change-vision.com/ja/product
[3] Yu Manta, Katsumi Wasaki. “Description and Consistency Checking of Distributed Algorithms in UML Models using Composite Structure and State Machine Diagrams”, Proc. of ITNG 2023, Springer AISC, vol.1445, pp.199-207, 2023.
[4] 萬田悠，和崎克己．“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”，情報処理学会 第 85 回全国大会 講演論文集, (5L-06), 289-290, Mar. 2023.