A-001

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

Description and Consistency Checking of Distributed Algorithms in UML Models using Composite Structure and State Machine Diagrams

萬田 悠 †　和崎 克己 ††
Yu Manta　Katsumi Wasaki

1　はじめに

システム開発の上流工程における自然言語による仕様記述は曖昧さを含み人的エラーが多くなる原因ともなる．これを防ぐ手段として形式手法がある．理論に基づいた形で仕様を厳密に記述し，上位設計での仕様の矛盾や一貫性を確認する，整合性検査を行うことで開発の早い段階でエラーを取り除き，手戻り，修正コストを削減する [1]．本研究では，UML（Unified Modeling Language）モデリングツールである astah*professional[2] が扱う，合成構造図とステートマシン図を用いて分散アルゴリズムを記述し，整合性検査を行う検査器を astah*プラグインに実装する．また，対象とする分散アルゴリズムのモデルは，モデル検査器 SPIN で使用されるモデル記述言語 PROMELA で記述される．PROMELA はプロセス間の通信や非決定的な振る舞いを記述することが可能な言語である．

2　UML を用いた PROMELA の記述

複数の 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．しかし，構造化クラスを定義しても，その構造化クラスがどのステートマシン図の振る舞いをするのかを明示しなければならない．そこで，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 (図 1)．

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，本研究で対象とするリーダー選挙問題の各プロセスが持つ固有の ID とその値の型を各構造化クラスにタグ付き値（value）と（type）をとして定義した (図 2)．

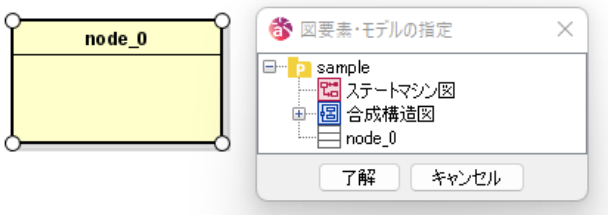

図 1　構造化クラスのハイパーリンク

図 2　ID の値と型のタグ付き値

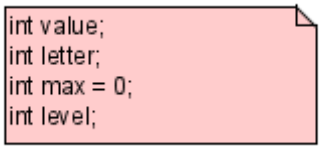

図 3　ノートへの記述の例

また，すべての 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 (図 3)．

2.3　通信チャネルの定義

PROMELA は各プロセスが相互に通信し，その通信イベントを元にモデルの振る舞いを記述する．本研究では提案手法として，従来の通信チャネルの記述に変更を加え，合成構造図で通信チャネルの記述を行った．従来の PROMELA での通信チャネルの定義と送受信は以下のように記述される．

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

† 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

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い，チャネルの 1 番目を用いて受信を行うことを表している．提案手法では送信動作を send，受信動作を receive として，各プロセスが送受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．

```
1 chan line[3] = [2] of {int}
2 #define port_01 1;
3 #define port_02 2;
4 #define send(port_ID,m) line[port_ID]!m;
5 #define receive(port_ID,m) line[port_ID]?m;
```

また，プロセスの内部で送受信動作を以下のように記述する．

```
1 send(port_01,value);
2 receive(port_02,value);
```

プロセスの送受信動作の遷移としては，図 4 のように，send の port_01 はマクロで定義された 1 より，line[1] のチャネルを使用し，value という int 型の値を送信することを表している．また，receive の port_02 はマクロで定義された 2 より，line[2] のチャネルを使用し，value という int 型の値を使用することを表している．

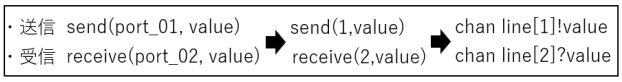

図 4　送受信動作の遷移

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　通信チャネルの記述

合成構造図でのチャネルの記述は構造化クラス，ポート，依存を用いて行う（図 5）．モデルのプロセスのインスタンスを表した構造化クラスに送信ポート（P_out）と受信ポート（P_in）を設置し，その送信ポートと受信ポートを依存（line）で繋ぐことで，PROMELA の通信チャネルとした．依存の矢印の向きはチャネルの通信方向と一致している．また，各依存（line）にはタグ付き値で値が設定されており，各構造化クラスはその値を基に自身の送信ポートと受信ポートがどのチャネルを使用するか定義する．図 6 の例では line1 の依存にタグ付き値の値 1 が設定されているので node_0 は node_1 に value を送信する際，chan line[1] を使用する．同様にして，node_2 は node_3 に value を送信する際は，line2 の依存のタグ付き値の値 2 から chan line[2] を使用する．

PROMELA での通信チャネルの実態は送信バッファと受信バッファの受け渡しであり，既存のチャネルの記述で，送受信動作を合成構造図で表現することが困難であった．そのため，チャネルの定義と送受信動作に変更を加えることで，合成構造図で PROMELA のチャネルを疑似的に表現した．

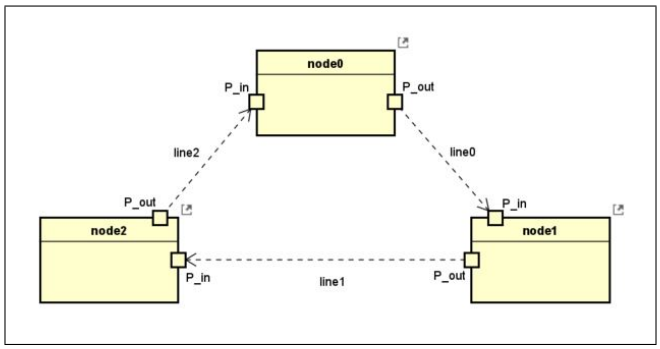

図 5　合成構造図を用いたチャネルの記述

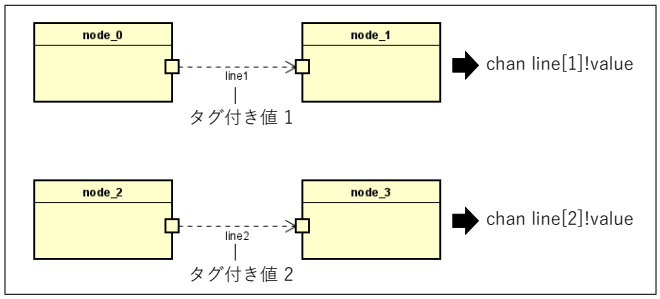

図 6　チャネルとタグ付き値

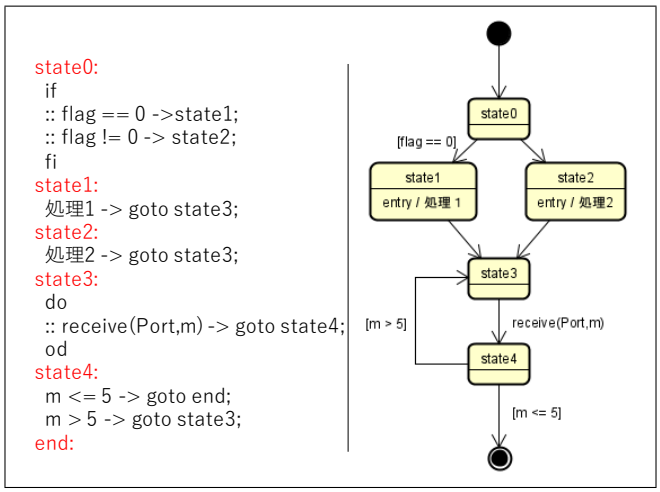

図 7　ステートマシン図を用いた振る舞いの記述 (右側)

2.5　プロセスの振る舞い

各プロセスの振る舞いは各クラスにハイパーリンクされたステートマシン図に記述する．図 7 は PROMELA の振る舞いをステートマシン図を用いて記述したものである．PROMELA ではプロセスの内部の処理にラベルを付け，goto 文で強制的に遷移させることが可能である．そのため，ステートマシン図における開始状態以外の状態をラベルとして扱い，トランジションを goto 文として扱う．また，if 文の条件文はトランジションのガード条件に，反復処理を行う do 文を使用した受信待機はトランジションのトリガーに記述する．

3　UML を用いたリーダー選挙問題のモデル記述

3.1　リーダー選挙問題の概要

本研究で対象とするリーダー選挙問題の概要を述べる．リーダー選挙問題は分散システムにおいて，唯一の

代表者を決定するモデルであり，基本的な動作としては，各プロセスは個別に ID を持ち，その ID を別のプロセスに送信する．その後，ID を受信した側は，受信した ID と自身の ID を比較し，より大きい ID をまた別のプロセスに送信する．この動作を片方向のリングネットワーク上に行い，自身の ID が一周したプロセスが代表者となる．下記にリーダー選挙問題（Chang Roberts）の疑似コードを示す [3].

Algorithm 1 Chang Roberts

```
status ← waiting
maxID ← 0 /*最大値の初期設定*/
myID /*自分の ID の値*/
     (「開始」が到着した時の動作)
if status = waiting then
    send (myID) to left
    maxID ← myID /*最大値を自分の ID に設定する*/
    status ← candidate /*リーダー候補者になる*/
end if
     (右隣からメッセージ（ID) を受信したときの動作)
if (ID = myID) and (status = candidate) then
                /*自分が候補者で，自分の ID が 1 周する*/
    status ← leader /*リーダーになる*/
else
    if (status = waiting) or (ID > myID) then
                          /*開始前か，大きな ID が届いた*/
        status ← notleader
                          /*リーダー候補者ではなくなる*/
    end if
    if ID > maxID then
                           /*受信した ID が最大値の場合*/
        maxID ← ID /*最大値の更新*/
        send (ID) to left /*左隣へ送る*/
    end if
end if
```

3.2 合成構造図を用いた記述

合成構造図を使用して，リーダー選挙問題のインスタンスとチャネルを記述する．図 8 は作成した合成構造図である．リーダー選挙問題では，参加者の中から代表者を 1 名決定するが，参加者は代表者が決定したことを知ることができなければ，永久に送信と受信を行ってしまい，プログラムがライブロックする原因となる．そのため，代表者が決定したことを，すべての参加者に伝達するブロードキャストプロセスを作成した．リーダー選挙問題で通常の送受信を行う，参加者のプロセスを 3 つ (node_0，node_1，node_2) 記述し，ブロードキャストを行うプロセス (broadcast) を，図の中心に記述した．各構造化クラスには送信ポート（P_out）と受信ポート（P_in）があり，ポートとポートをチャネルの方向に合わせて依存で繋げた．また，それぞれの構造化クラスには振る舞いとなる，ステートマシン図がハイパーリンクされている．

3.3 ステートマシン図を用いた記述

次にステートマシン図を使用して，リーダー選挙問題の振る舞いを記述する．図 9 は作成した参加者の振る舞いを記述したステートマシン図である．リーダー選挙問題の疑似コードにある，「開始」が到着したとき

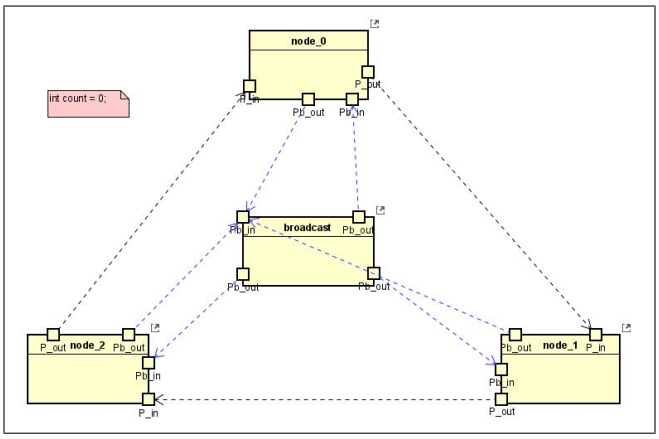

図 8　作成した合成構造図（参加ノード数 3）

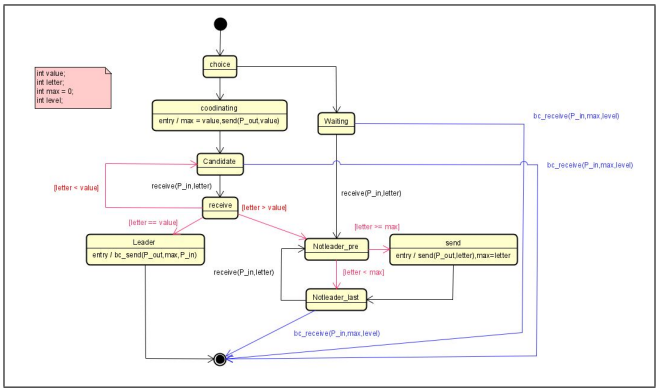

図 9　作成した参加ノードのステートマシン図

の動作を表現するために，「choice」という状態を開始状態の遷移先に記述し，参加者になるかどうかを非決定的に決定する．参加する場合は「coodinaiting」へ遷移し，「Candidate」で受信待ちを行う．参加しない場合は，「Waiting」へ遷移する．その後，1 度受信すると「Notleader_pre」に遷移し，受信待ちを行う．また，「Candidate」,「Waiting」,「Notleader_last」の状態でブロードキャストプロセスからの受信を行うと，終了状態へ遷移する．本研究では，参加者の振る舞いが同一であったので，用意するステートマシン図も 1 つにし，同じステートマシン図を共有させた．しかし，これでは各プロセスが利用するチャネルも共有してしまうので，ステートマシン図を解析する際に，ハイパーリンク元の構造化クラスのポートの ID を取得し，ステートマシン図のポートが記述された箇所を ID に書き換えた．

図 10 は作成したブロードキャストの振る舞いを記述したステートマシン図である．ブロードキャストでは，決定した代表者から ID と使用しているチャネルの情報を「stand」で受信し，「starting」で，ループと条件文を用いて，代表者以外の参加者に受信した ID を送信している．送信が終わると終了状態へ遷移する．

4 合成構造図とステートマシン図の整合性検査

合成構造図とステートマシン図で行う整合性検査の内容を検討した．合成構造図での整合性検査を行う項目は 16 種類，ステートマシン図での項目は 9 種類存在し，その一部を表 1 に示す．ナンバーは整合性検査の内容を管理しやすくするために検査を行う項目に数字を割り当

表 1　合成構造図とステートマシン図の検査項目の例

エラーレベル	ナンバー	不整合内容
Critical	CS-01	プロジェクトに合成構造図が 1 つも存在しない
Error	CS-05	合成構造図にポートが 1 つも存在しない
Error	CS-13	すべての依存にタグ付き値 (type) が 1 つも存在しない
Error	SM-05	ある状態に遷移先が存在しない
Warning	SM-08	ステートマシン図にノートが存在しない

表 2　合成構造図に基づくステートマシン図の検査項目

エラーレベル	ナンバー	不整合内容
Error	CS-SM-01	構造化クラスのポートの名前と一致するものがステートマシン図に 1 つも存在しない
Error	CS-SM-02	構造化クラスのタグ付き値である value がステートマシン図で使用されていない
Warning	CS-SM-03	構造化クラスの依存に定義された型がステートマシン図のノートに書かれていない

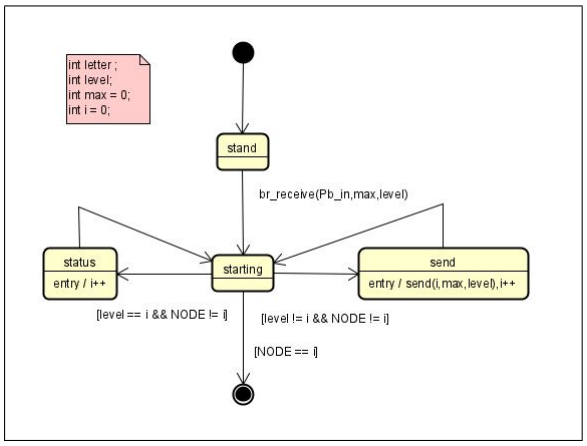

図 10　作成したブロードキャストのステートマシン図

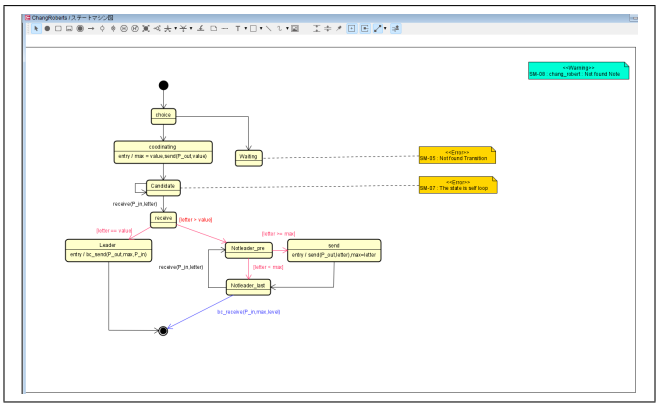

図 11　ノートの出力の例

て，エラーレベルは，検査内容を Critical（コード生成不可になるようなもの），Error（コンパイル時にエラーになるようなもの），Warning（それら以外の例外的なもの）の 3 つに分類したものである．検査器では，これらの項目をプロジェクト内のすべての合成構造図とステートマシン図で検査し，不整合が存在しなければ，検査した合成構造図の情報を基にそれぞれのステートマシン図で再度，整合性検査を行う（表 2）．

5　astah*プラグインによる検査器の実装

astah*professional 上へプラグインによる整合性検査器の実装を行った．検査器の開発言語には Java を使用し，作成した図の解析は astah*のモデルデータを活用するための Java インターフェース群である astah*API を用いて行った．記述したソースコードの行数はおよそ 1600 行である．検査器のフィードバックは，エラーレベルが Critical のものはウィンドウの画面に直接出力され，それら以外の検査結果は図上へのノート出力で行われる．合成構造図の不整合やステートマシン図の複数の要素に対する不整合は図の右上に出力されるが，ステートマシン図の不整合に関連する要素が 1 つの場合は，その要素に紐づく形でノートとノートアンカーが出力される (図 11)．また，ユーザーが作成したノートと区別するため，出力されるノートにはエラーレベルに応じて色付けを行い，作成した要素同士の距離が近い場合に出力されるノート同士が重ならないようにノートの位置を調整した．ユーザーは，このノートを確認しながら不整合箇所を修正し，修正された内容が記述されたノートは再度，検査器を実行すると削除される．

6　まとめと今後の課題

UML モデルである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自動コード生成を行う．本研究では，その段階の 1 つとして，合成構造図とステートマシン図を用いたリーダー選挙問題の記述を行い，それらを対象とした整合性検査を行うための検査器を astah*professional 上へ実装した．今後の課題としては，実際に作成した UML 図から PROMELA のコードを生成し，astah*professional 上から，モデル検査器 SPIN を起動した後，取得した反例ファイルを再び astah 上へフィードバックする機能を astah のプラグインへ追加する．また，UML 図の記述対象をルーティングプロトコルの一つである OSPF（Open Shortest Path First）などの，現在のリーダー選挙問題より規模が大きく，複雑なモデルに変更する．

参考文献

[1] 畑瀬尚之，和崎克己：“自動コード生成を目的としたテンプレートベースによる UML 上位設計の作成と整合性検査”，FIT2020（第 19 回情報科学技術フォーラム）講演論文集, (A-001), 45-46, 2020
[2] 株式会社チェンジビジョン：astah*professional, http://astah.change-vision.com/ja/product
[3] 真鍋義文：分散処理システム，森北出版株式会社，2013