B-020

UML 記述の仕様から SPIN モデル検査用 PROMELA モデルへの自動変換

Automatic Conversion from the Specification on UML Description to PROMELA Model for SPIN Model Checker

宮本 直樹 †　　和崎 克己 ††
Naoki Miyamoto　Katsumi Wasaki

1　はじめに

近年，ソフトウェア開発は大規模化，複雑化の一途を辿っている．そのため，ソフトウェアの仕様や振る舞いを人手で検証するのが困難になってきた．そこで，ソフトウェアの仕様を検証する手法として，モデル検査が注目されている．モデル検査とは，検査の対象となる仕様の振る舞いにおいて，仕様に含まれない不正な状態を，モデルが初期状態からとりうる状態を自動的に網羅することにより調べる技術である．モデル検査を行うことで，設計段階での欠陥を検出することができ，ソフトウェアの信頼性や安全性を向上させることができる．

現在利用できるモデル検査ツールは多数存在するが，本研究では SPIN モデル検査器 [1,2,3] を採用する．SPIN は，分散システムの形式的な検証を行うためのソフトウェアパッケージとして広く用いられている．SPIN でモデル検査を実行するには，PROMELA(PROcess Meta Language) 言語を用いて，検査対象のモデルを記述する．

本研究では，UML 図 [4] で書かれた上流モデルと要求仕様を，SPIN モデル検査器で使用する PROMELA 言語コードと時相論理式による仕様に自動変換する方法について検討する．まず，対象オブジェクトについて既存の UML ツールを用いて，複数のステートマシン図と，ステートマシンのインスタンスならびに通信チャネルの関係を記述した配置図としてエントリする．ここで，UML のステートマシン図には記述の曖昧性が存在するため，特にトランジションと終了状態に関する自由度を制限し，自動変換が容易となるようにした．UML のデザインエントリ全体を，XML 形式でエクスポートされたファイル経由で，PROMELA 言語コードへ自動変換する．また，UML 記述に仕様パターン [5] を埋め込むことで，線形時相論理式 (LTL) へ展開する機能も実現した．自動変換後の PROMELA コードと時相論理式をセットで扱うことで，UML からモデル検査器までの一貫した設計検証が可能となった．

2　UML と SPIN モデル検査器

検証対象のモデルをを，UML のステートマシン図と配置図で記述する．記述したモデルを，PROMELA に変換し，SPIN でモデル検査を実行する (3 で詳述).

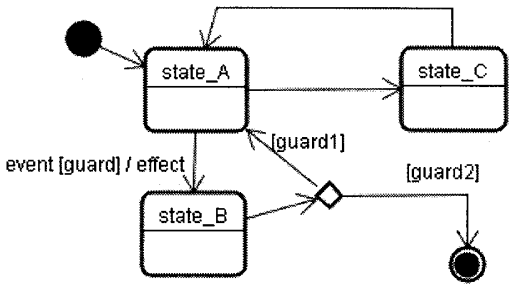

図 1　UML 記述: ステートマシン図の例

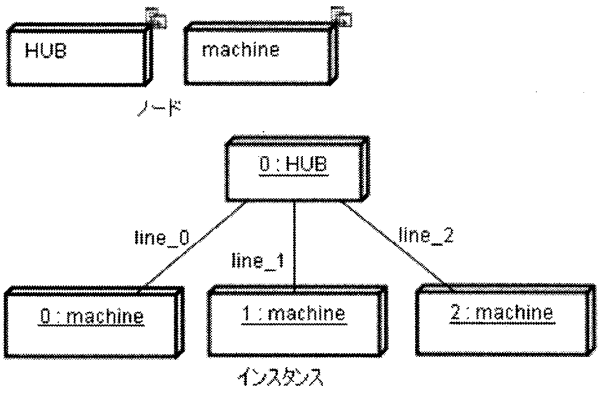

図 2　UML 記述: 配置図の例

2.1　UML

UML は，1995 年に Rational software 社が発表した仕様記述言語である．2004 年には UML の上位仕様である UML2.0 が発表された．UML2.0 以降では，13 種類の図が定められている．UML は 13 種類の図を，モデルの静的側面や動的側面といった用途に応じて使い分け，システムをモデリングする．オブジェクト指向システムの開発の際は，UML はクラス図やオブジェクト図などがよく使用される．本研究では，13 種類の図の中のステートマシン図と配置図を使用し，システムをモデリングする．

2.1.1　ステートマシン図

ステートマシン図は，インスタンスの変化を状態遷移として表現する図である。そのため，システムの動的な状態をモデリングできる．図 1 に，状態遷移の例を表す．現在の状態が state_A の時，外部から event を受け取り，条件 guard を満たすと，effect アクションを発生させて，現在の状態が state_B に遷移する．黒い丸で表されているのは開始状態であり，システムが最初に取る状

† 信州大学大学院工学系研究科,
Graduate School of Science and Technology, Shinshu University.
†† 信州大学工学部情報工学科,
Department of Information Engineering Faculty of Engineering, Shinshu University.

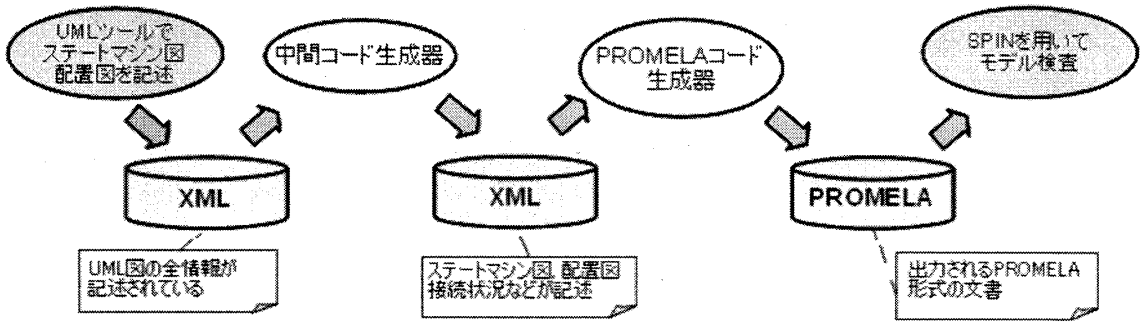

図 3　UML から PROMELA モデルへの変換の流れ

態を表す．白い丸の中に黒い丸が含まれているのは終了状態である．現在の状態が，終了状態に遷移すると，システムは終了となる．菱形で表されるのは選択状態である．選択状態からは複数本のアークが出力される．出力アークにはそれぞれ guard コマンドが定義されており，選択状態に遷移すると，各出力アークの guard コマンドを評価し，条件が真となるアークが 1 本選択され，選択されたアークで遷移が起きる．

2.1.2　配置図

配置図は，システムに用いられる物理的な要素が，どのように配置されているか記述する図法である．例えば，インスタンス間の通信路の接続状況や，インスタンスの物理的な数などを記述する．

図 2 は配置図の記述例である．配置図では，定義したノードからインスタンスが生成できる．インスタンスは「インスタンス名:ノード名」と，下線を引いて定義する．図 2 では，ノード HUB からはインスタンス名が 0，ノード machine からはインスタンス名が 0,1,2 のインスタンスが生成される．各インスタンス間の接続状況は，リンク線を用いて記述する．図 2 では，machine のインスタンス 0,1,2 が，それぞれ，line_0,line_1,line_2 のリンク線で，HUB のインスタンスと接続される．

2.2　SPIN モデル検査器

SPIN は，ベル研究所で G.J.Holzmann 博士を中心に開発されているモデル検査ツールである．1980 年代から研究が進められており，実際の開発現場でも広く普及している．SPIN の特徴として，分散システムや通信プロトコルなどの，相互通信のシステム検証に使われることが多い．

2.2.1　PROMELA

PROMELA は，並行して動作するプロセスや，非決定的な振る舞いなどを容易に記述できる仕様記述言語である．PROMELA は，並行動作するプロセス毎に仕様を記述する．SPIN は，PROMELA で記述されたモデルのランダムシミュレーションの機能を有する．また，SPIN は，PROMELA で記述されたモデルを検査できる．SPIN で検査できる内容は，表明検査，デッドロック検査等の形式検証も実行できる．

2.2.2　線形時相論理 (LTL) 式

時相論理とは，通常の論理式に時間の経過という概念を加えた論理体系である．時相論理により，命題が「ある時点では真だが，いつかは偽になる」といった時間の経過で変化する論理式を記述することができる．モデル検査で，モデルの性質を検証したい場合，多くのモデル検査器では時相論理を用いて性質を記述する．SPIN は，線形時相論理 (Linear Temporary Logic : LTL) 式を採用している．LTL は，時間の進み方が一直線の性質を記述する．LTL は，命題論理式に時相演算子と呼ばれる演算子を追加して，論理式を記述する．LTL 式で検証したい論理式を記述することで，SPIN は，PROMELA で記述されたモデルが，その論理式を受理するかどうかを自動で検証する．

3　UML 記述から仕様記述言語 PROMELA への自動変換

UML に記述されたモデルを，仕様記述言語 PROMELA へ自動変換する．

3.1　UML 記述の適用範囲

今回，UML でモデルを記述する際は，ステートマシン図と配置図に限定して記述する．モデルの振る舞いはステートマシン図で記述する．モデルのインスタンス数や，インスタンス間の接続状況等は配置図で記述する．

3.2　自動変換ツールの構成

図 3 に，UML から PROMELA モデルへの自動変換の流れを示す．変換器は，Perl 言語で記述した．変換の流れは，まず，検証したいモデルを，UML のステートマシン図と配置図で記述する．UML 記述ツールは，株式会社チェンジビジョンの astah* Professional[6] を採用した．このツールは，記述した UML を XML 形式のファイルとして出力する機能を有する．この XML ファイルには，自動変換に不必要な項目も含まれているため，余分な項目を削除した XML 形式の中間コードを生成する．中間コード生成器の概要は，まず，XML ファイルを読み込み DOM 形式でパースする．パースした XML は木構造となり，再帰的に走査できる．走査中に余分な項目があった場合，削除する．走査終了後，XML ファイルを再構築し，中間コードとして出力する．

次に，中間コードを，PROMELA 形式の文書に変換する．変換手順の概要を述べる．まず，中間コードを入力し，DOM 形式でパースする．この XML ファイルには，状態，アーク，インスタンス，チャネルなど，PROMELA への変換に必要な項目が含まれている．これらの情報を Perl の機能を用いて，ハッシュに格納する．木構造の

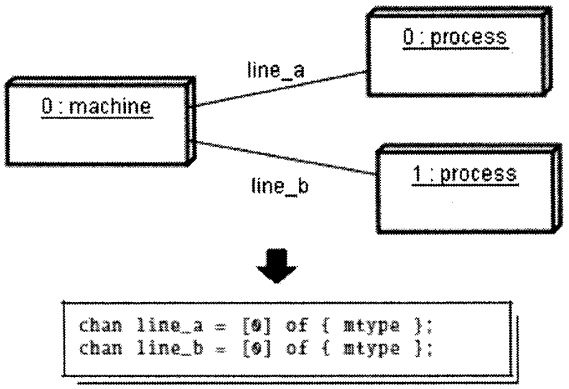

図 4　配置図での通信用チャネルの定義

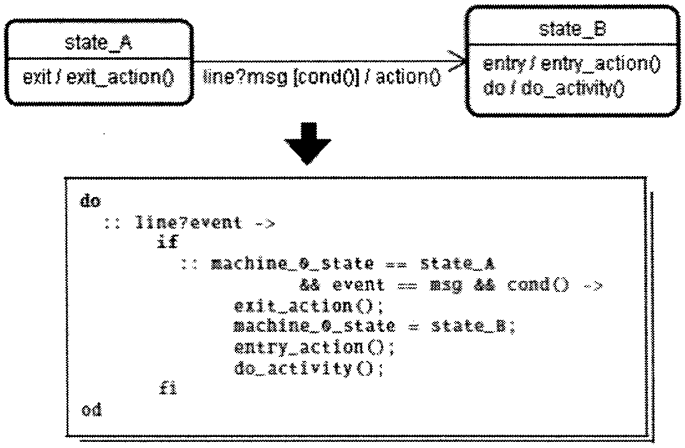

図 5　ステートマシン図から PROMELA への変換

走査が終了すると，ハッシュには PROMELA への変換に必要な項目が格納されるため，ハッシュを走査して，PROMELA 形式の文書に変換する．

3.3　自動変換の流れ

3.3.1　プロセスの定義

PROMELA は，プロセス毎に振る舞いを記述する．プロセスの数は，UML の配置図で定義したインスタンスの数とする．そのため，まず配置図でノード，インスタンスを定義する．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．各プロセス間の通信用チャネルの定義は，配置図で行う．なお，通信用チャネルは同期通信とする．図 4 は，配置図での通信用チャネルの定義例である．図の上部は，ノード machine のインスタンスが，ノード process のインスタンス 0 と，インスタンス 1 に接続される．それぞれのインスタンスは，line_a，line_b のリンク線で接続される．この配置図を変換すると，図 4 の下部の PROMELA でのチャネルの宣言文となる．

3.3.2　状態の定義

ステートマシン図の状態遷移を，PROMELA に変換するため，状態変数を定義する．状態変数には，現在のステートマシン図の状態が格納される．状態遷移によって現在の状態も変化するので，状態変数の値も変化する．本研究では，文献 [2] で定義された，状態遷移の順序を採用する．

1. 入力イベントの実行
2. 遷移条件の真偽を確認
3. 現在の状態の exit ラベルの実行
4. 遷移上のアクションの実行
5. 状態遷移をし，次の状態に移る
6. entry ラベルの実行
7. do ラベルの実行

状態遷移の順序は，上記の繰り返しとする．また，状態遷移のための入力イベントは必ず<チャネル名>?<メッセージ>といった，PROMELA でのチャネルの受信の定形とする．図 5 は，ステートマシン図から PROMELA への変換例である．ステートマシン図は，現在の状態が state_A の時，チャネル line がメッセージ msg を受信すると，状態遷移が発生する．

3.3.3　コンポジット状態の定義

ステートマシン図では，状態の内部に，新たな状態列が入れ子になっているコンポジット状態を記述できる．遷移した先の状態がコンポジット状態だった場合，コンポジット状態内の状態列を新たに処理する．そのため，コンポジット状態内の状態列がどこから始まるか，一意に決まらなければならない．コンポジット状態内の初期状態は必ず存在し，かつただ一つの状態を指すとする．UML では，コンポジット状態の内部の状態列からコンポジット状態の外部への遷移，もしくは外部の状態から内部への遷移が記述できるが，これらの表現も曖昧な定義であり，複数の意味解釈ができるので本研究では採用しない．状態の遷移は，同じ階層内の状態列にのみ遷移できるとする．

4　UML 記述への仕様パターン埋込と LTL 式への展開

SPIN は，LTL 式で検証したいモデルの性質を記述することで，モデル検査を行う．文献 [5] は，モデル検査でよく使われる検査式を，仕様パターンとして論理式へ変換する手法を提案している．ここでは，文献 [5] で提案されている仕様パターンに基づき，UML に埋め込まれた仕様パターン記述を，LTL 時相論理式へ自動展開する手法について説明する．

4.1　仕様パターンの埋込

UML にはノート機能が搭載されており，UML の図中に注釈を付加できる．ノート機能に，検証したい仕様パターンを記述し，検証に必要な状態やインスタンスと関連付ける．ノート機能に，$$...$$で囲まれた記述があった場合，仕様パターンとする．仕様パターンの埋込みは，ステートマシン図，配置図で行う．1 つのプロセスでの仕様パターンの記述には，ステートマシン図を用い，複数のプロセスでの仕様パターンの記述には，配置図を用いる．配置図に，仕様パターンを埋め込む場合は，インスタンス名を指定する必要がある．

4.2　LTL 時相論理式への展開

予め，仕様パターンと，変換後の LTL 式の対応表を作成しておき，Perl の機能を用いて，対応表をハッシュに

読み込む．UML のノート機能に記述された仕様パターンのパターン名を読み込み，ハッシュから，そのパターン名と対応する LTL 式を取得する．ここで，LTL 式の論理変数を定義する必要がある．論理変数は，ステートマシン図では，UML のノート機能から関連付けられる状態と連携する．配置図では，記述された仕様パターンに指定されたインスタンス名から論理変数を定義する．論理変数の定義は，SPIN では，define 文を用いて宣言する．LTL 式と，論理変数の定義を，テキスト形式で出力する．これらの記述を，SPIN に入力することで，要求仕様の検査を自動的に実施可能となった．

5 変換例

本研究で試作した変換ツールを適用したものとして，HD/DVD レコーダのネットワーク越しコピー問題 [2] を例として挙げる．各プロセスの接続関係と，プロセスの振る舞い記述を UML で作成し，性質を検証する．

HD レコーダは，TV 番組の録画と保存，DVD レコーダは，DVD への番組保存の機能をそれぞれ有する．HD レコーダに録画した番組を DVD にコピーする場合を考える．DVD に番組を保存するには，HD レコーダに保存された番組のデータを，ネットワークを介して DVD レコーダに送る．DVD レコーダは，送られた内容を DVD にコピーすることで実現する．

DVD レコーダは，自リソース確保後，HD レコーダにリソース確保要求を送信する．DVD レコーダは，HD レコーダのリソース確保確認を受信すると，HD レコーダに再生確保要求を送信する．HD レコーダは，再生確認を DVD レコーダに送信した後，コンテンツの再生を開始する．DVD レコーダは，再生されているコンテンツの内容を DVD にコピーする．

この振る舞いの，ステートマシン図と，配置図をそれぞれ図 6，図 7 に示す．配置図のノード HD は HD レコーダ，ノード DVD は DVD レコーダを表している．今回，検査する性質は，「DVD レコーダの状態がコピー中状態になる前に，HD レコーダの状態が，待機状態から再生中状態になる」とする．配置図に，ノート機能を用いて，仕様パターンを埋め込んだ．これらの UML 図を変換器に入力し，PROMELA モデルと，LTL 式に出力した．出力された LTL 式を図 8 に示す．

生成された PROMELA モデルと LTL 式を用いて，SPIN でモデル検査を実行すると，SPIN は valid を出力する．以上により，仕様パターンで定義した性質について，対象モデルがその仕様を満足することをモデル検査器によって確認することができた．

6 まとめ

UML 図で記述されたモデルを，変換器を通して PROMELA モデルに一貫して自動変換することで，PROMELA のコードや，複雑な時相論理式を記述することなく，SPIN でモデル検査が実行できる．

UML 図を PROMELA に変換する際，同期通信チャネルを採用した．しかし，実際のシステムでは，バッファを介する非同期通信チャネルを使用する場合が多い．非同期通信モデルから，PROMELA モデルへの変換例は，文献 [2] に掲載されている．非同期通信を対応させ，更

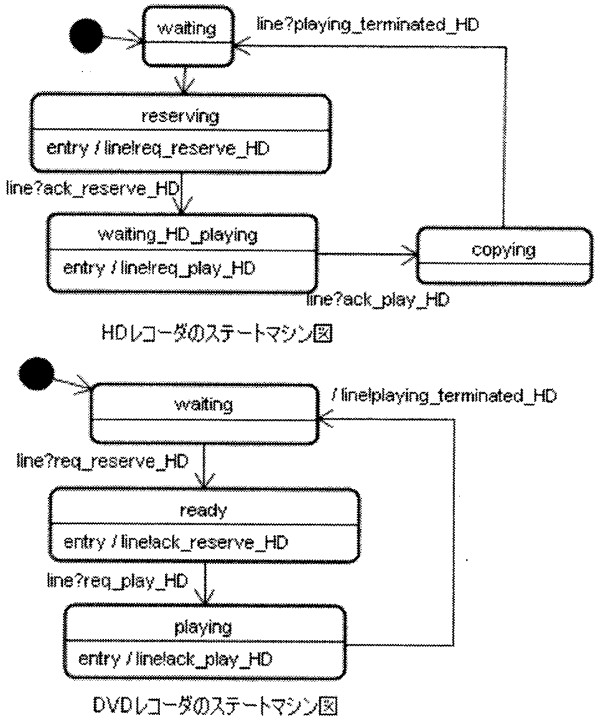

図 6 HD レコーダ，DVD レコーダのステートマシン図

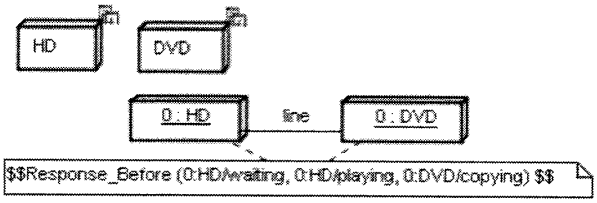

図 7 HD レコーダ，DVD レコーダの配置図

```
<>r -> (p -> (!r U (s && !r))) U r

#define s ( HD_0_state == waiting )
#define p ( HD_0_state == playing )
#define r ( DVD_0_state == copying )
```

図 8 変換された LTL 式

に複雑なモデルの変換を実現したい．

UML には，時系列でのモデルの動作を記述できるシーケンス図がある．シーケンス図で記述されたモデルは，要求仕様の LTL 式へ変換できる．今後は，シーケンス図から，LTL 式への自動変換も検討したい．

参考文献

[1] Gerard J.Holzmann, “THE SPIN MODEL CHECKER”, Addison-Wesley, 2004.
[2] 吉岡信和 青木利晃 田原康之, “SPIN による設計モデル検証”, 近代科学社, 2009
[3] 中島震, “SPIN モデル検査”, 近代科学社, 2008.
[4] 児玉公信, “UML モデリング入門”, 日経 BP 社, 2008.
[5] SPEC PATTERNS, http://patterns.projects.cis.ksu.edu/documentation/patterns.shtml.
[6] 株式会社 チェンジビジョン, http://www.change-vision.com/.