ISSN 0913-5685
信学技報(IEICE Technical Report) Vol. 111 No. 308

電子情報通信学会技術研究報告

IEICE Technical Report

SWIM2011-18 – SWIM2011-29

ソフトウェアインタプライズモデリング

Software Interprise Modeling

2011 年 11 月 18 日

November 18, 2011

社団法人 電子情報通信学会
The Institute of Electronics, Information and Communication Engineers
http://www.ieice.org/

電子情報通信学会技術研究報告目次

CONTENTS

〔ソフトウェアインタプライズモデリング〕

〔Software Interprise Modeling〕

—提案型エンタプライズモデリング　ワークショップ—

上記目次において論文番号に＊印がついている論文は、査読を経て採録された論文である。
その他は査読対象外である。

Note：＊：The papers marked with* at its number in the table of Contents have been accepted and printed with reviewing by the SWIM Program Committee.

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SWIM2011-19(2011-11)

上流設計からモデル検査プロセスまでの一貫設計検証環境

～UML 記述から SPIN モデル検査器用プロセス定義及び線形時相論理式への自動変換手法～

宮本　直樹†　　和崎　克己††

† 信州大学大学院工学系研究科　〒 380–8553 長野県長野市若里 4–17–1
†† 信州大学工学部情報工学科　〒 380–8553 長野県長野市若里 4–17–1
E-mail: †10ta544a@shinshu-u.ac.jp, ††wasaki@cs.shinshu-u.ac.jp

あらまし　SPIN モデル検査器を実行するには，専用の仕様記述言語 PROMELA で対象モデルを記述する．また，検査対象の仕様の記述には線形時相論理 (LTL) 式を用いる．本研究では，システム開発の上流設計段階で使われる UML 図を，PROMELA コード及び LTL 式に自動変換する手法を提案する．UML のステートマシン図と配置図を組みで利用し，ステートマシン図に，対象モデルの振る舞いや配置状況，要求仕様を記述し，PROMELA コードへの自動変換を実現する．一方，要求仕様として，UML のシーケンス図を仕様パターンに準拠した記法に制限・展開することで，LTL 式の自動生成を実現する．以上の機能を有する援用ツール群を試作し，ある通信プロトコルを対象とした上位設計に対する自動変換を実施し，評価を行った．

キーワード　上流設計, モデル検査, UML, SPIN, PROMELA, 線形時相論理式, 仕様パターン

An Integrated Design and Verification Environment from Upstream Design to Model Checking Process

-Automatic Conversion from UML Descriptions into the Process Definitions and Linear Temporal Logic for SPIN Model Checker-

Naoki MIYAMOTO† and Katsumi WASAKI††

† Faculty of Engineering, Graduate School of Science and Technology, Shinshu University.
4-17-1 Wakasato, Nagano city, Japan
†† Faculty of Engineering, Shinshu University.　4-17-1 Wakasato, Nagano city, Japan
E-mail: †10ta544a@shinshu-u.ac.jp, ††wasaki@cs.shinshu-u.ac.jp

Abstract　To execute a SPIN model checker, the targeted model has to be described by the dedicated specification description language “PROMELA”. Also, linear temporal logical (LTL) expressions are used for the description of the specification to be tested. In this study, we propose a method for automatically transforming the UML diagram used at the upstream design stage of system development to PROMELA codes and LTL expressions. Automatic transformation to PROMELA codes is achieved by the following procedures: (1)Combine the state machine chart and layout drawing of UML; and (2) Describe the behavior, layout condition, and requested specifications of the targeted model to the state machine chart. Meanwhile, as a requested specification, by restricting and expanding the sequence diagram of UML to a notation complying with the specification pattern, automatic generation of LTL expressions is achieved. We made computer-aided tools that have the functions above experimentally, and conducted evaluations of this method based on a case study of design verification.

Key words　Upstream Design, Model Checking, UML, SPIN, PROMELA, Linear Temporal Logic, Specification Patterns

1. はじめに

近年，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 [1,2,3]．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，実適用例が多く代表的な SPIN ツールは，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，専用の仕様記述言語 PROMELA(PROcess MEta LAnguage) を用いて，検査対象のモデルを記述する．SPIN は，PROMELA で記述されたモデルが，線形時相論理 (Linear Temporal Logic : LTL) 式で記述された要求仕様を満たすことを網羅的に検査する．

上流工程におけるモデルの記述法から，モデル検査器用のプロセス定義へ自動変換する手法は，広く研究されている．UML のコラボレーション図を，オートマトンとして変換し，SPIN で性質を検証する手法 [4] や，シーケンス図とステートマシン図を用いてモデルの振る舞いを記述する手法 [5]，クラス図とステートマシン図で対象モデルを記述する手法 [6] 等が挙げられる．

本研究では，UML 図のステートマシン図，配置図，シーケンス図で記述された上流モデルと要求仕様を，SPIN モデル検査器向けのプロセス定義，及び LTL 式へ自動変換する手法について検討する．プロセス定義への自動変換については，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．LTL 式への自動変換については，シーケンス図に対象となる要求仕様を記述する．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汎用交換ファイル形式経由で，PROMELA 言語コード及び LTL 式へ自動変換する．自動変換後の PROMELA コードと LTL 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

2.1 UML

UML [7] は，OMG(Object Management Group) により管理されている仕様記述言語である．本研究では，UML で定義された 13 種類の図の中で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，モデルの状態変化を状態遷移として表現する図である。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．黒い丸で表示されるシンボルは開始状態，白い丸の中に黒い丸が含まれているシンボルは終了状態を表す．菱形で表されるのは選択点である．選択点からの出力アークにはそれぞれ guard コマンドが定義されており，選択点に遷移すると，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発生する．

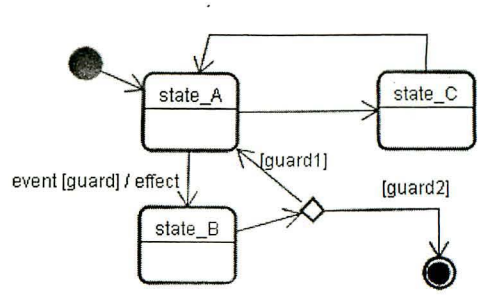


図 1　UML 記述: ステートマシン図の例

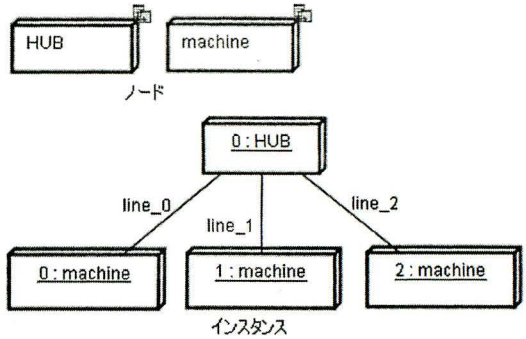


図 2　UML 記述: 配置図の例

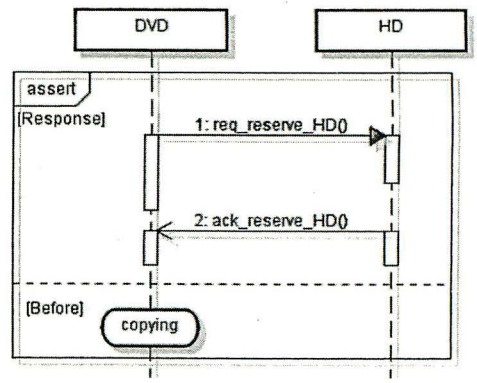


図 3　UML 記述:シーケンス図の例

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の，配置状態を記述する図法である．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．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．

2.1.3 シーケンス図

シーケンス図は UML，対象モデルの時系列での動的な流れを記述できる．記述例を図 3 に示す．また，複合フラグメントを用いることで，モデルの制御構造を記述できる．複合フラグメント要素は，内部に新たな複合フラグメント要素を記述することで，入れ子の構造が記述できる．

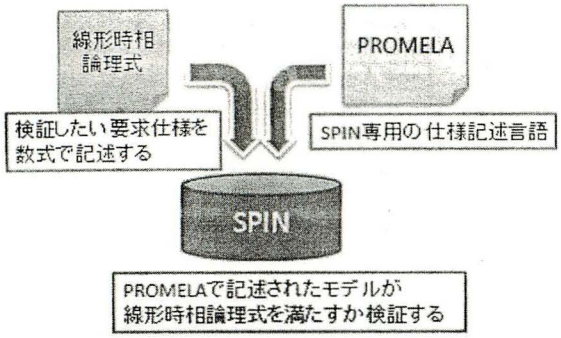


図 4　SPIN モデル検査の概要（入力ファイル群）

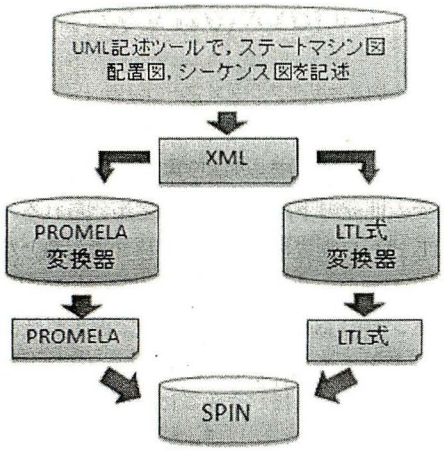


図 5　UML から SPIN 用プロセス記述及び LTL 式への変換

2.2　SPIN モデル検査器

SPIN は，モデル検査ツールの一種である．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．モデル検査の概要を図 4 に示す．

SPIN でモデル検査を実行する際は，専用の仕様記述言語 PROMELA を用いてモデルを記述する．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特徴を持つ．SPIN は，PROMELA で記述されたモデルの表明検査，デッドロック検査等の性質を検証できる．

2.3　線形時相論理式

時相論理 [8] 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．SPIN は，線形時相論理 (Linear Temporary Logic : LTL) 式を採用している．LTL 式で検査対象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．

2.4　仕様パターン

仕様パターン [9] は，Dwyer らが提唱する性質記述パターンである．これは，モデル検査の際によく用いられる要求仕様を予めパターン化して時相論理式として提供したものである．

3.　UML から SPIN 用プロセス定義への自動変換

UML から PROMELA コードへの自動変換の手法について提案する．UML で記述されたモデルを自動変換することで，上流工程における視覚的了解性を損なうことなく，対象モデルのモデル検査が実行できる．モデルの記述には，UML のステートマシン図と配置図を用いた．

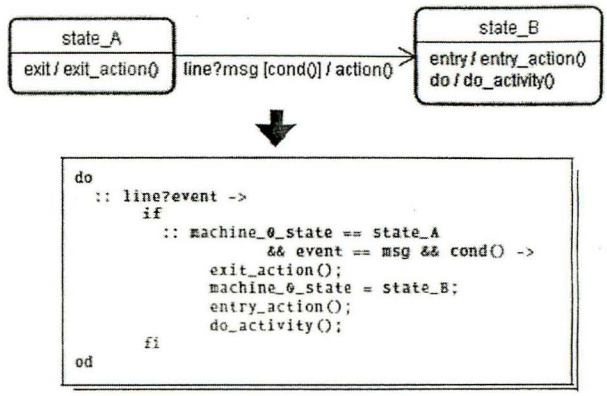


図 6　ステートマシン図から PROMELA への変換

3.1　自動変換の流れ

UML から PROMELA コードへの自動変換の流れを，図 5 の左側に示す．変換器は，Perl 言語で実装した．変換の流れは，まず，対象のモデルを，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 [10] 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．出力された XML ファイルの，自動変換に必要な項目を Perl のハッシュに格納する．ハッシュを走査して，PROMELA 形式の文書に変換する．

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．状態遷移時に，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．

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．これは，自動変換を完了させるためには，設計の上でメッセージをやり取りする型について確定しておく必要があるためである．図 6 は，ステートマシン図から PROMELA への変換例である．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．ステートマシン図から PROMELA をモデルへの具体的な変換例は，5.2 節に示す．

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，新たな状態列が存在するコンポジット状態も記述できる．本研究では，コンポジット状態の PROMELA コードへの変換を検討する．遷移先の状態がコンポジット状態だった場合，内部の状態列を新たに処理する．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．また，コンポジット状態の定義を厳密にするため，状態の遷移は，同じ階層内の状態列にのみ遷移

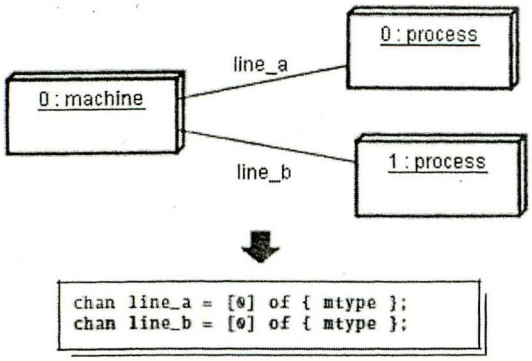


図 7　配置図での通信用チャネルの定義

できるとする．

3.4　通信用チャネルの定義

PROMELA では，振る舞いをプロセス毎に記述し，プロセス間の通信は，chan 型の変数で行われる．自動変換後のプロセスや，通信用チャネルは，UML の配置図で定義する．通常，配置図は下流工程に用いられる図だが，配置図に用いられるインスタンスやリンク線が PROMELA への自動変換に適していると判断し，配置図を採用した．配置図のリンク線を，chan 型の変数に変換する．配置図から PROMELA での chan 型の変数の定義の例を図 7 に示す．インスタンス 0:machine は，0:process と 1:process にそれぞれ line_a と line_b のリンク線で接続されている．これらのリンク線が，chan 型の変数の変換対象である．配置図から PROMELA モデルへの変換例は，5.2 節に示す．

4.　UML から線形時相論理式への自動変換

UML から LTL 式へ自動変換する手法について検討する．要求仕様の記述には，UML のシーケンス図を用いた．変換対象の指定や，インスタンス間の時系列での状態変化が記述できるため，シーケンス図を採用した．自動変換の具体的な例は，5.3 節に示す．

4.1　自動変換の流れ

シーケンス図で記述された要求仕様から，LTL 式への自動変換の流れを図 5 の右側に示す．3 節で述べた，ステートマシン図，配置図から PROMELA コードへの自動変換と同様に，UML 記述ツールは，astah* Professional を用いる．変換器は Perl 言語で記述した．UML 記述ツールでシーケンス図を記述し，ツールの機能として XML ファイルを出力する．自動変換に必要な項目を XML ファイルから抽出し，LTL 式に変換する．

4.2　シーケンス図の適用範囲

検証したい要求仕様をシーケンス図で記述する．記述された図を LTL 式へ自動変換するが，一意に変換するため，シーケンス図で記述する要素を限定する．用いる要素は，ライフライン要素，複合フラグメント要素，メッセージ要素のみとした．

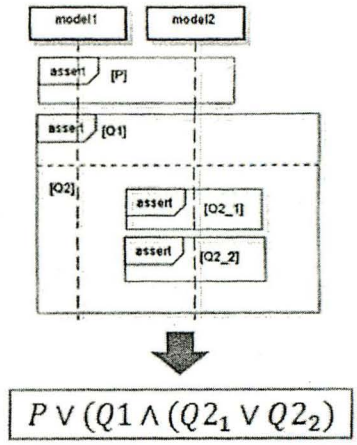


図 8　複合フラグメントの論理演算

4.3　複合フラグメント

複合フラグメントはモデルの制御構造を表す図法であり，通常は，システム開発の下流工程において用いられる．シーケンス図を，LTL 式へ自動変換するためには変換対象となる部位を指定する必要があり，本研究では複合フラグメントを採用し，複合フラグメントの assert フラグメントで囲まれた部位を LTL 式への変換対象とする．また，LTL 式へ一意に変換するため，変換する要求仕様は，仕様パターンに準拠した．

4.3.1　複合フラグメントの論理演算

複合フラグメントの組み合わせにより，論理演算を表現する．複数の複合フラグメントを離して記述した場合，それぞれの LTL 式を論理和演算子 ∨ で結合する．複合フラグメント内に複数のオペランドが記述されている場合は，それぞれの LTL 式を論理積演算子 ∧ で結合する．複合フラグメント内に，新たに複合フラグメントがあった場合は，内部の論理式を () で囲み，一つの数式として表現する．

複数のフラグメントの構造を論理演算に変換した例を図 8 に示す．複合フラグメント Q_1 と $Q2$ はオペランドで結合されているため，$Q1 \wedge Q2$ で表現される．また，P と $Q1$ 及び $Q2$ は離れて記述されているため，論理和演算で結合する．その結果，$P \vee (Q1 \wedge Q2)$ で表される．更に，$Q2_1 \vee Q2_2$ は，$Q2$ の内部にあるため，$Q2$ は $(Q2_1 \vee Q2_2)$ に書き換えられる．よって，全体の論理演算は $P \vee (Q1 \wedge (Q2_1 \vee Q2_2))$ で表される．

4.3.2　複合フラグメントと仕様パターンの対応

複合フラグメントのガード部位に仕様パターン名を記述し，対応する LTL 式を取得する．また，複合フラグメントに，before, after, until と記述することで，特定の範囲内での時間制約となるスコープを表現する．スコープは，仕様パターンで定義されている．before のみが追加された場合は，仕様パターンの Before スコープを表し，after のみが追加された場合は，After スコープを表す．before と after が共に記述された場合と，after と until が記述された場合は，それぞれ Between スコープと After_Until スコープを表す．

図 9 は，複合フラグメントを LTL 式に変換した例である．

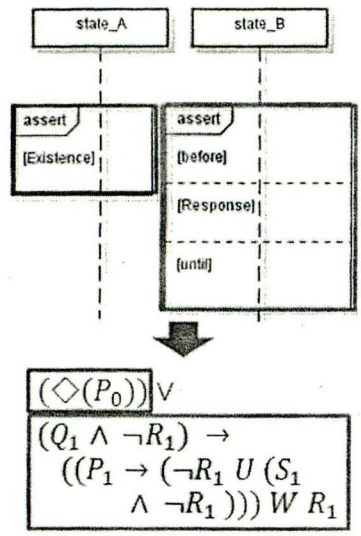


図 9　複合フラグメントと仕様パターン

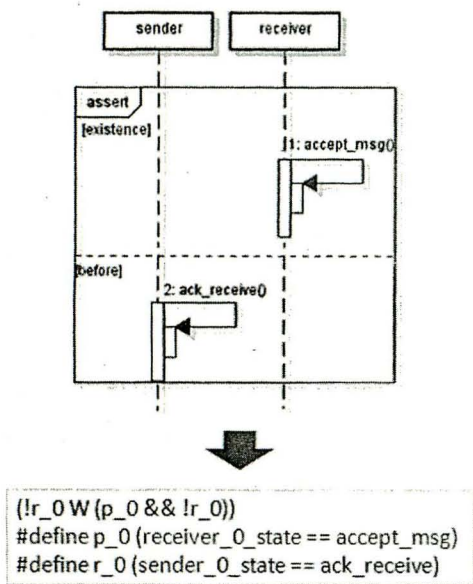


図 10　メッセージと論理変数の対応

上部の複合フラグメントのガード部位には existence が記述されている．これは，仕様パターンでの LTL 式は $\Diamond p$ である．下部の複合フラグメントは after，Response，until が記述されている．これは，After_Until スコープで成り立つことを表す．この表現を仕様パターンに対応付けて LTL 式に変換すると，$(Q \wedge \neg R) \rightarrow ((P \rightarrow (\neg R\ U\ (S \wedge \neg R))))\ W\ R$ となる．また，2 つの複合フラグメントは離れて記述されているため，それぞれの LTL 式は ∨ 演算子で結合される．

4.4　メッセージと論理変数の対応

シーケンス図のメッセージ要素を，モデルの状態値として論理変数に対応させる．シーケンス図のライフラインが，自分自身にメッセージを送信した場合は，そのメッセージに記述された内容を状態値とする．

図 10 に例を挙げる．この複合フラグメントは，before と existence で構成されており，この仕様パターンで用いられる論理変数は，P と R である．図 10 は，existence の内部は accept_msg，

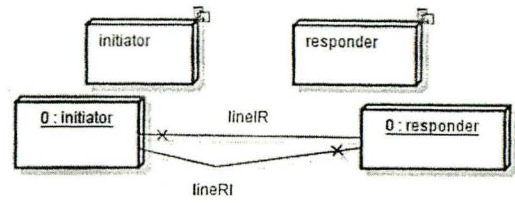
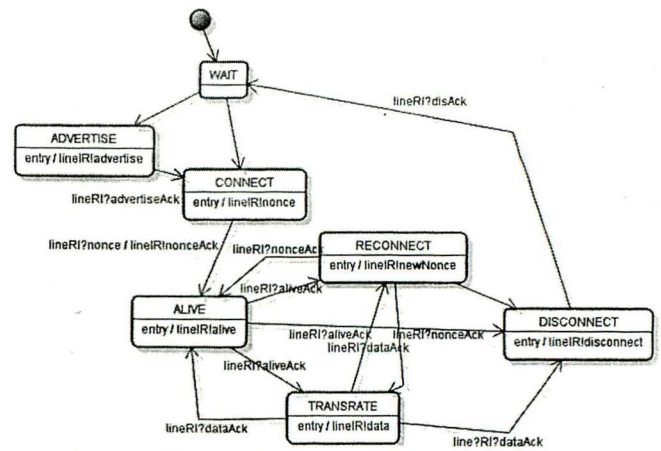
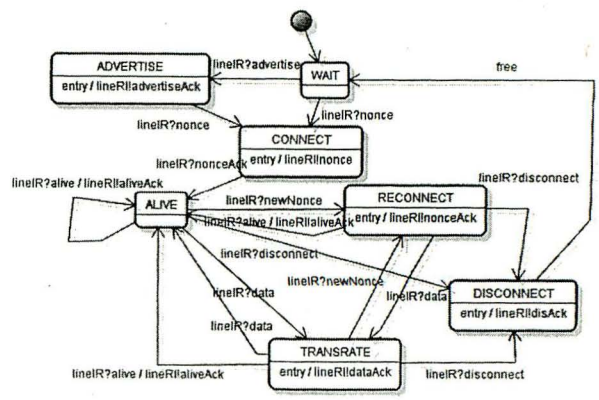


図 11　永続化プロトコルの配置図



(a) インスタンス Initiator のステートマシン図



(b) インスタンス Responder のステートマシン図

図 12　永続化プロトコルのステートマシン図

before の内部は，ack_receive がそれぞれ状態値となる．この状態値を，LTL 式で用いられる論理変数と対応させる．対応させる記述は，モデル検査器 SPIN で用いられる記法とした．図 10 の下部は出力された LTL 式である．

5.　変　換　例

5.1　永続化プロトコル

試作した変換器の適用例として，P2P 仮想ネットワークにおける移動体接続の永続化プロトコル [11] を挙げる．これは，移動端末間での通信において，接続が途絶えた場合，接続フェーズの最初から開始せず，それまで接続を行っていた種々の情報からノンス情報を生成し，それをノード間で確認・交換しあうことで再接続時のコストを抑えるプロトコルである．従来のプロトコルに ALIVE 状態と RECONNECT 状態を追加している．ALIVE 状態は，一定時間間隔で端末間で決まったメッセージを送り合うことで，接続を確認する状態である．RECONNECT

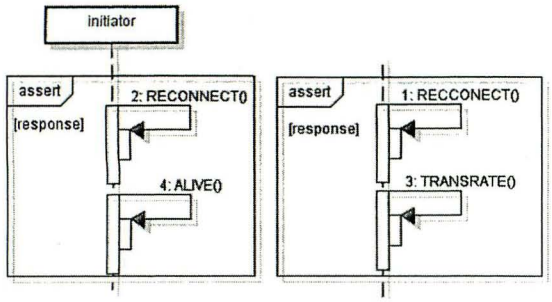


図 13　要求仕様のシーケンス図

```
([](p_0 -> <>s_0)) || ([](p_1 -> <>s_1))

#define p_0 (initiator_0_state == RECONNECT)
#define p_1 (initiator_0_state == RECONNECT)
#define s_0 (initiator_0_state == ALIVE)
#define s_1 (initiator_0_state == TRANSRATE)
```

図 14　変換された LTL 式

```
State-vector 40 byte, depth reached 173, errors: 0
      415 states, stored (485 visited)
      315 states, matched
      800 transitions (= visited+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.022 equivalent memory usage for states
              (stored*(State-vector + overhead))
    0.277 actual memory usage for states
              (unsuccessful compression: 1250.41%)
          state-vector as stored = 684 byte +
                                   16 byte overhead
    2.000 memory used for hash table (-w19)
    0.305 memory used for DFS stack (-m10000)
    2.501 total actual memory usage
```

図 15　検証結果

状態は，通信が途中に途切れた場合，再接続を行う状態である．

5.2　永続化プロトコルのステートマシン図・配置図記述

上述のプロトコルは，送受信を制御する Initiator と，データを受け取る Responder で構成される．それぞれインスタンスの配置状況を，図 11 に配置図で示す．図 12 に各インスタンスの振る舞いをステートマシン図で記述した図を示す．ステートマシン図及び，配置図を，試作した PROMELA コード変換器を用いて，PROMELA をモデルへ変換した．

5.3　要求仕様のシーケンス図記述

検証する要求仕様の一例は，「ノードの状態が RECONNECT 状態に遷移した場合，必ず ALIVE 状態または TRANSRATE 状態に遷移する」となる．要求仕様をシーケンス図で記述したものが図 13 である．今回の要求仕様を検証するには，インスタンス Initiator の応答性を検証すれば良いため，仕様パターンの Response を変換対象とした．シーケンス図上での複合フラグメント名を Response とした．また，要求仕様は 2 つの論理式の和となっているため，2 つの複合フラグメントを離して記述した．試作した LTL 式変換器を用いて，シーケンス図で記述した要求仕様を LTL 式に自動変換する．出力された LTL 式は図 14 である．

出力された PROMELA と LTL 式を用いて，SPIN モデル検査器でモデル検査を行った結果，永続化プロトコルが，要求仕様を満たすことを確認した．出力結果を図 15 に示す．Initiator プロセスは，再接続状態に到達した場合，ALIVE 状態もしくは TRANSRATE 状態へいつかは遷移することが保証された．検証に要した状態空間については，状態数 415，深さ 173 であった．

6.　まとめと今後の課題

UML による上流設計から，PROMELA コード及び LTL 式へ自動変換することで，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．これにより，言語コードや数式を記述することなく，容易にモデル検査が実行できる．また，UML を単一のツールで複数の図を統一して記述することにより，曖昧な記述ができる UML において，各図の整合性を取ることができる．

UML から，モデル検査用のプロセス定義へ自動変換する際，一意に変換するために，UML の記法を制限した．より複雑なモデルの記述に対応できるよう，他の記法も用いたいと考えている．本研究で提案する設計手法では，UML で対象モデルを記述するため，モデル検査の実行結果に反例が出力された場合でも，入力元の UML 図と照らし合わせることで，レビューが行える．今後は，反例結果を解析し，UML 記述ツールにフィードバックすることで，より視覚的な効果を合わせたレビューを支援する手法を検討したい．

謝辞　本研究の一部は科学研究費（23500174）の助成を受けたものである．

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，東京，2009.
[3] 中島震，SPIN モデル検査，近代科学社，東京，2008.
[4] T.Schafer, A.Knapp, and S.Merz, “Model Checking UML State Machines and Collaborations, ” Electric Notes in Theoretical Computer Science 47, 2001.
[5] 大貫智洋，上野浩一郎，磯田誠，“状態遷移を持つオブジェクト間通信のモデル検査技術，” 第 9 回情報科学技術フォーラム，2010.
[6] 八鍬豊，野田夏子，“UML モデルの振舞いのモデル検査における表現方法について，” 第 9 回情報科学技術フォーラム，2010.
[7] 児玉公信，UML モデリング入門，日経 BP 社，東京，2008.
[8] Z.Mannna, A.Pnueli, Temporal Verification of Reactive Systems: Safety, Springer, 1995.
[9] SPEC PATTERNS, http://patterns.projects.cis.ksu.edu/documentation/patterns.shtml.
[10] 株式会社 チェンジビジョン，http://www.change-vision.com/.
[11] 呉ヒョク，“P2P 仮想ネットワークにおける移動体接続の永続化プロトコル設計と検証，” 信州大学大学院修士論文，2010.
[12] 宮本直樹，和崎克己，“UML 記述の仕様から SPIN モデル検査用 PROMELA でモデルへの自動変換，” 第 9 回情報科学技術フォーラム，2010.
[13] 宮本直樹，和崎克己，“UML シーケンス図の構造記述から線形時相論理式への自動変換手法，” 第 10 回情報科学技術フォーラム，2010.