B-028

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

Automatic Conversion from the Structure Description on UML Sequence Chart to Linear Temporal Logic

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することができる．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，専用の仕様記述言語 PROMELA(PROcessMeta 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か検証できる．

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．本研究では，UML 図で書か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，特に LTL 式へ自動変換する手法について検討する．まず，形式化したい要求仕様を，既存の UML ツールのシーケンス図に記述する．この時，シーケンス図の複合フラグメント要素内に記述された要求仕様のみを，LTL 式への変換対象とした．複合フラグメント要素に仕様パターン名を記入し，LTL 式と要求仕様を一意に対応付けることで，自動変換を実現する．さらに，複数の複合フラグメント要素を組み合わせることで、LTL 式同士の論理演算を実現した．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，LTL 式及び，PROMELA 言語コードへ自動変換する．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[6] 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

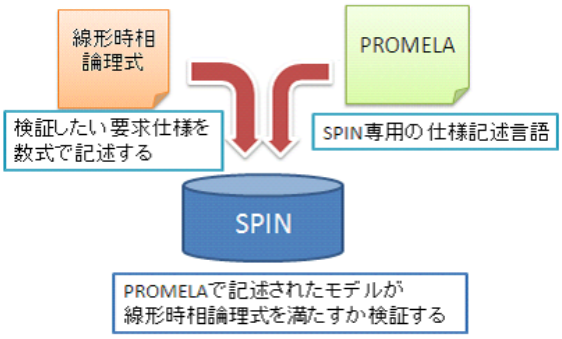

図 1　SPIN モデル検査器

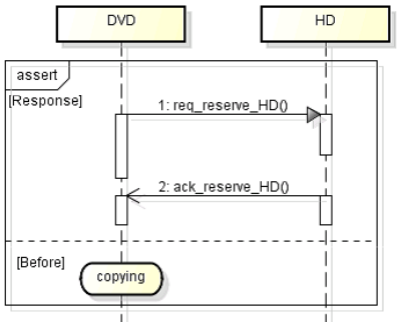

図 2　UML 記述:シーケンス図の例

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．

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．

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．

† 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.

2.3 線形時相論理式

線形時相論理 (Linear Temporary Logic：LTL) 式は，時相論理式の一種であり，時間の進み方が一直線の性質を記述する．時相論理とは，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．

モデル検査器 SPIN は，PROMELA で記述されたモデルが LTL 式で記述された要求仕様を満たすか検証する機能を有する (図 1)．しかし，複雑な要求仕様を LTL 式で記述する場合，数式も複雑になってしまう問題点がある．

3 シーケンス図と LTL 式

検証対象の要求仕様を，UML のシーケンス図で記述する．記述したモデルを，仕様パターンに対応させて LTL 式に変換する．変換の具体的な手順は 4 節で詳述する．

3.1 シーケンス図

シーケンス図は UML で定義された図法の 1 つである．シーケンス図は，対象モデルの時系列でのメッセージの流れを記述できる．また，複合フラグメント要素を用いることで，ループやクリティカルセクションといったモデルの制御構造を記述できる．本研究では，シーケンス図を用いて検証対象の要求仕様を記述する．

3.2 仕様パターン

仕様パターン [7] とは，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．パターン化されている時相論理式は，LTL 式や，計算木論理 (Computational Tree Logic: CTL) 式などがある．仕様パターンを論理演算で組み合わせることで，更に多様な要求仕様を記述できる．

4 シーケンス図から LTL 式への自動変換

シーケンス図で記述された要求仕様を LTL 式へ自動変換する．

4.1 UML から PROMELA モデルへの自動変換

著者らは，UML から PROMELA モデルへの自動変換する手法について検討している [4]．UML で記述されたモデルを自動変換することで，視覚的かつ容易に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．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り，システムの動的な状態をモデリングできる．自動変換したい対象モデルの振る舞いを，状態遷移を用いて動的に記述する．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．配置図を用いて，対象モデルの各インスタンス間の配置状況や，通信路の接続状況などを記述する．また，これらの図法に，要求仕様の性質を埋め込むことで，LTL 式に自動変換する手法も提案した．

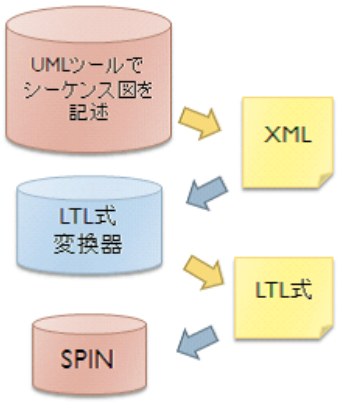

図 3　シーケンス図から LTL 式への変換の流れ

本稿では，UML のシーケンス図に記述された要求仕様を LTL 式へ自動変換する手法を提案するが，出力される LTL 式は [6] の変換器と合わせて用いることを目的としている．そのため，自動変換後の LTL 式の記法は [6] の変換器の規則に則っている．

4.2 自動変換ツールの構成

図 3 に，シーケンス図で記述された要求仕様から，LTL 式へ自動変換する流れを示す．まず，検証したい要求仕様を UML のシーケンス図で記述する．UML 記述ツールは，株式会社チェンジビジョンの astah* Professional[8] 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 ファイルを解析し，LTL 式として出力する．解析する際は，XML ファイルを DOM 形式でパースした．パースした XML ファイルは木構造として解析され，再帰的に走査できる．再帰的な走査により，複数の要素が入れ子の構造になっているモデルの変換が可能となる．

4.3 シーケンス図記述の適用範囲

検証したい要求仕様は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．ライフライン上に，検証したい要求仕様を時系列で記述する．複合フラグメントは，シーケンス図の制御構造を記述する際に用いる要素である．

4.4 複合フラグメント

複合フラグメントの一種である assert で囲まれた部位を LTL 式への自動変換の対象とする．本来，assert フラグメントは，ガード条件が成り立たなければ処理ができない構造を記述できる複合フラグメントである．

4.4.1 複合フラグメントの論理演算

シーケンス図の複合フラグメント要素の組み合わせにより，論理演算を表現する．複数の論理式を組み合わせることで，要求仕様の記述性の向上を図る．

複合フラグメントで囲った部位が LTL 式の適用範囲

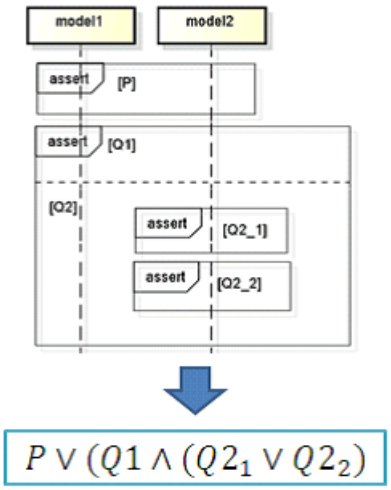

図 4　複合フラグメントの論理演算

である．複数の複合フラグメントを離れて記述した場合，それぞれの LTL 式を論理和演算子 ∨ で結合する．複合フラグメントは，同一の要素内に複数のオペランドを追加できる．複数のオペランドで接続された場合は，それぞれの LTL 式を論理積演算子 ∧ で結合する．また，複合フラグメントは，要素内に新たな複合フラグメントを記述できる．要素内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，1 つの数式として表現する．

図 4 は，複数のフラグメントの構造を論理演算に変換した例である．複合フラグメント P，$Q1$，$Q2$ で構成されており，$Q2$ の内部には更に複合フラグメントが記述されている．この時，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，$Q2$ の内部には，$Q2_1 \vee Q2_2$ が記述されている．この演算式は，$Q2$ 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．

4.4.2　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する．この名前を読み込んだ際に，対応する LTL 式を取得する．また，複合フラグメントに追加されたオペランドに，before，after，until と記述することで，ある特定の範囲内で要求仕様が成り立って欲しい場合の記述ができる．before のみが追加された場合は，仕様パターンの Before スコープを表す．これは，「命題 Q が真になる前」の範囲である．after のみが追加された場合は，After スコープを表す．これは，「命題 Q が真になる前」の範囲である．また，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．Between スコープは，「命題 Q が真になり，命題 R が真になる間」の範囲であり，After_Until スコープは，「命題 Q が真になり，命題 R が真になるまでの間」の範囲である．Between スコープは，命題 R が真にならない限り真にはならないが，After_Until スコープは，命題 Q が真である限り，命題 R の真偽を問わず真となる特徴を持つ．

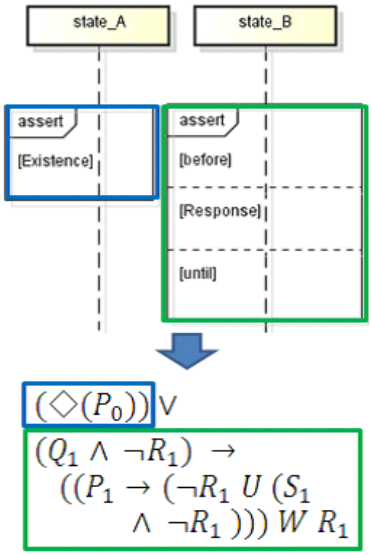

図 5　複合フラグメントと仕様パターン

図 5 は，複合フラグメントを LTL 式に変換した例である．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，「いつかは命題 p が真になる」を表しており，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 3 つのオペランドから構成されている．ガード部位はそれぞれ after，Response，until が記述されている．Response は，「命題 P は，必ず命題 S に到達する」を表している．また，after，until のオペランドで囲まれているため，After_Until スコープで成り立つことを表し，この複合フラグメントは「命題 Q が真であり，命題 R が真になるまでに，命題 p は，必ず命題 S に到達する」を意味している．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 つの命題は複数のフラグメントで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．

シーケンス図を用いることで，視覚的に要求仕様が記述できる．また，複雑な LTL 式も容易に出力できる．

4.5　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に記述された内容を，モデルの状態として論理変数に対応させる．図 6 に対応させる例を挙げる．この複合フラグメントは，オペランド before と existence で構成されており，「命題 P は，命題 R が真になる前に必ず真になる」を意味する．即ち，この時相論理で用いられる論理変数は，P と R である．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 6 では，オペランド existence の内部は，receiver ライフラインがメッセージ accept_msg を自分自身に送信している．また，オペランド before の内部では，sender ライフラインが，メッセージ ack_receive を自分自身に送信している．よって，accept_msg と 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 6 の下部は出力された LTL 式であ

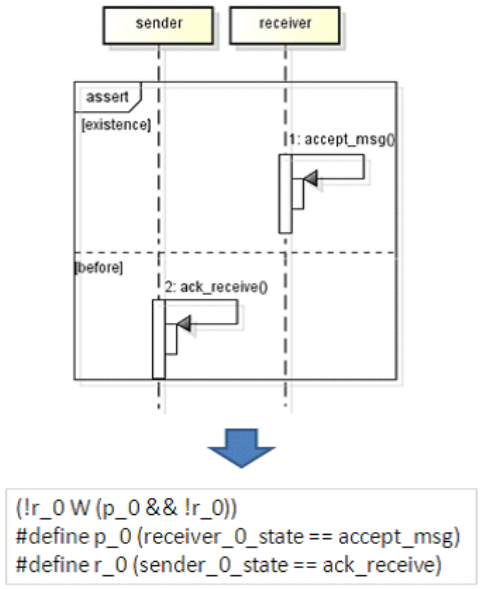

図 6　メッセージと論理変数の対応

る．この要求仕様は，「送信者が確認メッセージ受信状態 (ack_receive 状態) に遷移する前に，必ず受信者はメッセージ受信状態 (accept_msg) 状態に遷移する」を LTL 式に変換したことになる．

5　変換例

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 [5] 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い，確認することで，接続が成り立っていることを確認する状態である．RECONNECT 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．今回検証する要求仕様は，「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」とする．

上述のプロトコルを，UML のステートマシン図で記述した図が図 7 である．記述した図を，[4] で試作した変換器を用いて，PROMELA モデルへ変換した．また，要求仕様をシーケンス図で記述した図が図 8 である．変換器を用いて，要求仕様を LTL 式に自動変換する．出力された LTL 式は図 9 である．SPIN モデル検査器を用いて，永続化プロトコルが，要求仕様を満たすことを確認した．

6　まとめ

シーケンス図で記述された要求仕様を，LTL 式に自動変換することで，複雑な数式を記述することなく出力できる．また，シーケンス図の複合フラグメント要素を組み合わせることで，視覚的にかつ多様な記述性を実現した．UML 図から PROMELA コードへの自動変換器と合わせて用いることで，モデル検査が容易に実行できる．

シーケンス図から LTL 式へ一意に変換するために，記法を制限した．今後は，UML 図の他の記法も用いて，更に記述性の向上を図りたい．

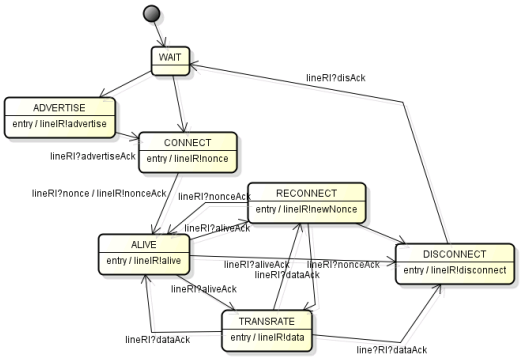

図 7　永続化プロトコルのステートマシン図

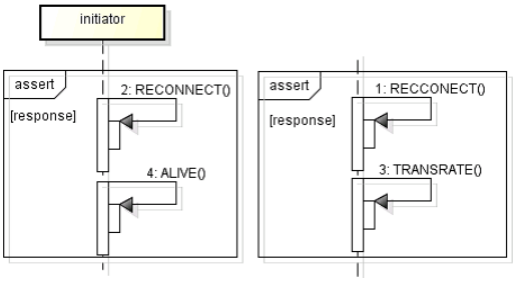

図 8　要求仕様のシーケンス図

```
([](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 9　変換された LTL 式

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] 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”，情報科学技術フォーラム，2010.
[5] 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.
[6] 児玉公信, “UML モデリング入門”, 日経 BP 社, 2008.
[7] SPEC PATTERNS, http://patterns.projects.cis.ksu.edu/documentation/patterns.shtml.
[8] 株式会社 チェンジビジョン，http://www.change-vision.com/.