B_003

シーケンス図と状態遷移図で記述されたUMLモデルを対象としたモデル検査による形式的検証

Formal verification for UML Sequence diagrams and Statecharts using model checking.

佐藤　貞仁† 宮崎　仁† 横川　智教† 佐藤　洋一郎† 早瀬　道芳†
Sadahito Sato Hisashi Miyazaki Tomonori Yokogawa Youichiro Sato Michiyoshi Hayase

1. まえがき

UMLはオブジェクト指向のソフトウェアシステムのためのモデリング記法であり，並行システムを始めとした様々なシステムをモデル化するために用いられている．しかしながら，設計されたモデルが設計者の意図した通りに動作することは保証できないため，モデルに検証を行い，正しく動作するか否かを確かめることが必要となる．形式的に記述されたシステムを自動的に検証する手法として近年注目されているのがモデル検査である．本論文では，UMLのシーケンス図と状態遷移図を用いて記述されたソフトウェアシステムの仕様を対象として，システムが意図された通りに設計されているか否かをモデル検査手法を用いて自動的に検証する手法を提案する．

2.UMLによる仕様記述

図1はソフトウェアのWatchdog mechanism[1]をモデル化したシーケンス図である．シーケンス図はオブジェクト間のメッセージのやり取りを時系列で表したものであり，イベントの発生順序を視覚的に表すことができる．矩形はオブジェクトを表している．矩形から縦方向に伸びる線はオブジェクトの活性区間を表している．矢印はメッセージを表し，閉実線矢じりの矢印は手続き的な同期メッセージを，開き矢印は非同期なメッセージを表している．

図2は図1のapplicationクラスに対する状態遷移図である．状態遷移図では丸みつきの矩形は状態，矢印は遷移を表している．遷移にはイベント/アクションという形式のラベルがつけられている．イベントは遷移を発火させるためのトリガであり，アクションは遷移が発火した際に行われる処理である．

3. 記号モデル検査

本研究では，記号モデル検査を用いて検証を行う．記号モデル検査では状態空間や遷移関係を論理関数を用いて表現し，モデル検査を論理関数の処理として実現する．また，検証すべき性質は時相論理の一種である計算木論理（CTL）を用いて記述される．

CTLでは，通常の命題論理に「将来いつかある性質が成り立つ」,「次の時刻である性質がある性質が成り立つ」等の時間に関係する性質を記述するための演算子を付け加えた論理体系である．時相論理とは，全称記号 A，存在記号 E のいずれかと，「次の時刻」を意味する X，「将来いつか」を意味する F,「将来的に」を意味する G,「～まで」を意味する U を含む4種類の時相論理演算子を組み合わせて性質を表現することができる．

本研究では状態遷移図で記述された仕様からシステムの遷移関係を表す論理式を生成し，シーケンス図で記述

†岡山県立大学　情報工学部

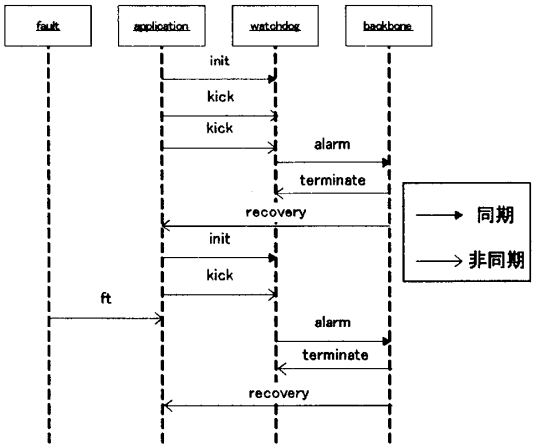

図 1: シーケンス図（WatchDog mechanism）

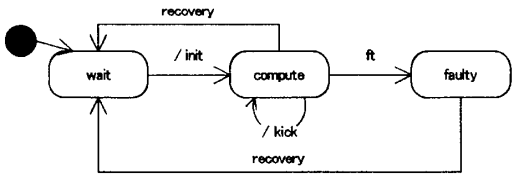

図 2: 状態遷移図（application）

されたメッセージの順序をCTLで表現する手法を提案することで，システムが与えられたシナリオ通りに動作するかを検証する．

4. システムの遷移関係を表す論理式表現

ここでは記号モデル検査を行うために，状態遷移図で記述された仕様からシステムの遷移関係を表す論理式を生成する手法について述べる．まず，イベント，アクション，状態をそれぞれ二値変数として表す．ここでは例として，図2を用いて説明する．まず各イベントとアクションを二値変数として表現する．例えばイベントrecoveryは二値変数 $recovery$ として表す．同様にしてアクションinitは二値変数 $init$ として表す．また，オブジェクトの状態は変数 $S_{Application} \in \{wait, compute, faulty\}$ を用いて表す．

以上のように求めた変数を用いて状態遷移図の各遷移を論理式で表現する．例えば，状態 $faulty$ から状態 $wait$ への遷移は，状態 $faulty$ でイベント $recovery$ が生起しているときに発火可能であり，遷移の発火後は状態 $wait$

へとなりイベントが破棄されるため，この遷移の発火は以下のような論理式で記述される．

$$S_{Application} = faulty \wedge recovery$$
$$\wedge S'_{Application} = wait \wedge \neg recovery'$$
$$\wedge init' = init \wedge ft' = ft \wedge kick' = kick$$

ここで変数 $S'_{Application}$，$recovery'$ はそれぞれ変数 $S_{Application}$，$recovery$ の次の状態での値を表している，また，$init' = init \wedge ft' = ft \wedge kick' = kick$ と記述することで，この遷移において他の変数の値が変化しないことを表している．同様にしてすべての遷移を論理式により表現し，論理和で結合することで，システム全体の遷移関係を表す論理式表現を求めることができる．

5. メッセージの順序の表現

ここでは，シーケンス図で記述されたシナリオから，メッセージが順序どおりに送受信されているかを表す論理式表現を求める手法について述べる．まず図 3 に示すように，各メッセージに MS1，MC1 というようなラベルをつける．MS1 は 1 番目のメッセージが送信されたことを示し，MC1 は 1 番目のメッセージが消費されたことを示している．これらのラベルを用いてシーケンス図のシナリオを表す状態遷移モデルを生成する．このモデルの状態は変数 $CP \in \{MS1, MC1, \cdots, MC12, END\}$ を用いて表す．状態 $CP = END$ であるときシナリオの終了，つまりすべてのメッセージがすべて送信され，消費されたことを表している．

例えば，ラベル MS1，MC1 がついたメッセージ init が送信され，メッセージが消費されるまでのシナリオを表現する論理式は次のように書くことができる．

$$CP = MS1 \wedge init \wedge CP' = MC1$$
$$\vee$$
$$CP = MC1 \wedge \neg init \wedge CP' = MS2$$

ここでは簡単のため，遷移により変化しない変数は省略している．また，システムの初期状態では $CP = MS1$ であるとする．同様に，シーケンス図中のすべてのメッセージについても論理式を生成し，論理和で結合することでシナリオ全体の順序を表す論理式表現を求めることができる．そして，このシナリオの順序を表す論理式と状態遷移図の遷移関係を表す論理式とを論理積で結合することで，シーケンス図のシナリオに沿った状態遷移モデルを求めることができる．

6. 検証する性質

ここでは，時相論理式（CTL）によって，どのように検証する性質を表現するかを述べる．ここで，検証する性質とは，状態遷移図で記述されたシステムがシーケンス図により記述されたシナリオを満たすかどうかである．

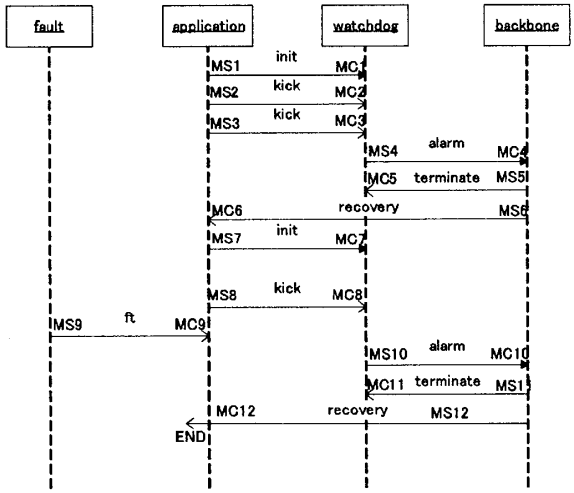

図 3: メッセージにラベルを付けたシーケンス図

シナリオを表す時相論理式は前節の変数 CP を用いて次のように書くことができる．

$$AGEF\,((CP = MS1) \rightarrow EF\,(CP = MC1)$$
$$\rightarrow EF\,(CP = MS2) \rightarrow EF\,(CP = MC2)$$
$$\vdots$$
$$\rightarrow EF\,(CP = MC10) \rightarrow EF\,(CP = MS11)$$
$$\rightarrow EF\,(CP = MC11) \rightarrow EF\,(CP = END))$$

システムの遷移系列がこの時相論理式を満たすならば，システムはシーケンス図で記述されたシナリオを満たすことがわかる．

7. モデル検査による検証

提案法を用いて Watchdog mechanism の仕様を論理式で記述し，モデル検査ツール SMV を用いて検証を行った．検証の結果，この状態遷移図により記述されたシステムの仕様が，シーケンス図により記述されたシナリオを満たしていることが確認された．

8. まとめ

本研究では UML のシーケンス図と状態遷移図を用いて記述された仕様を論理式に変換し，SMV を用いて検証を行う手法を提案した．今後の課題としては，他の UML 図で記述された仕様も含めた検証を行う手法を検討中である．

参考文献

[1] S.Bernardi, et al. , "From UML Sequence Diagrams and Statecharts to analysable Petri Net models, " Proceedings of the third international workshop on Software and performance (WOSP 2002), Rome, Italy, pp.35-45, ACM Press, 2002.