定理証明に基づくプロトコルの設計支援

福澤 俊幸* 中島 俊介** 長谷川 晴朗*
*沖電気工業 (株) **沖通信システム (株)

1 はじめに

通信システム等のプロトコルのモデル化の一つに状態遷移図に基づいた方法がある. 実際, プロトコルの仕様記述段階では, 多くの状態遷移図が作成され, それらを基にプログラムの設計は行われている. 従って, 状態遷移図段階での機械的な検証は, 非常に重要である. 本論文は, プロトコルの意味検証にモデルチェックアルゴリズムを利用した検証法とその過程で得られる証明木の利用方法を報告する.

2 プロトコルのモデル

本論文で検証の対象となるプロトコルは, 下に記すような状態遷移図によって表現されているものとする.

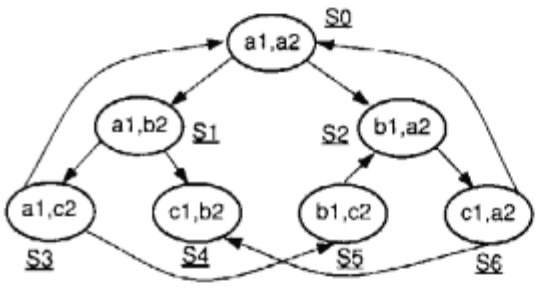

図 1: プロトコル仕様のモデル例

3 プロトコルの意味検証

3.1 モデルチェッカ

状態遷移図で表現されたプロトコルは, 各状態で成立する条件を命題論理式とみなす事により時相論理の解釈に利用されている, Kripke の可能世界モデルと同一の構造を持つ [3]. その構造は, 3 項組 M=(S,R,P) で表現されている. ここで, S は状態の有限集合, R は状態の遷移関係を表す状態の二項関係, そして P は各状態で成立する論理式を表している.

Protocol Design Support based on Theorem Proving
Toshiyuki FUKUZAWA*, Shunsuke NAKAJIMA**,
Haruo HASEGAWA*
*Oki Electric Industry Co., Ltd.
**Oki Telecommunication Systems Co., Ltd.

時相命題論理は, 古典命題論理に時間に関するオペレータを加えることにより定義される. 追加する時相オペレータの種類により様々な時相論理が提案されている. 本稿では, 以下の規則で構成される分岐時相論理を対象に説明を行う [1].

1. 古典命題論理式は, 時相論理式である.
2. p,q が, 時相論理式であるならば, ¬p, p∧q, EXp, AU(p,q), EU(p,q) も時相論理式である. 特に, $AX(\mathrm{p}) \equiv \neg EX(\neg \mathrm{p})$, $AF(\mathrm{q}) \equiv AU(\mathrm{true},\mathrm{q})$, $EF(\mathrm{q}) \equiv EU(\mathrm{true},\mathrm{q})$, $AG(\mathrm{p}) \equiv \neg EF(\neg \mathrm{p})$, $EG(\mathrm{p}) \equiv \neg AF(\neg \mathrm{p})$ と記す.

可能世界モデルにより, 各時相オペレータは以下のように解釈される.

$s_0 \models EX(p)$ | $(s_0,t) \in R$ なる t が存在して $t \models p$

$s_0 \models AU(p,q)$ | 全ての経路 $(s_0,s_1,\ldots)$ に対し, $i \geq 0$ が存在して, $s_i \models q$ かつ $j(0 \leq j < i)$ に対し $s_j \models p$

$s_0 \models EU(p,q)$ | ある経路 $(s_0,s_1,\ldots)$ に対し, $i \geq 0$ が存在して, $s_i \models q$ かつ $j(0 \leq j < i)$ に対し $s_j \models p$

モデルチェッカとは, 与えられた時相論理式が与えられた可能世界モデル上で成立することを判定するアルゴリズムである [1][2].

3.2 検証の手続き

モデルチェッカをプロトコルの意味検証に利用するには, 検証するプロトコルの仕様を時相論理式で表現し, その"論理式の否定"に対してモデルチェックを行う. プロトコルが仕様を満たしていることを検証するには, 状態遷移図がその対応する論理式の可能世界モデルには成らないことを証明すればよい.

例として, 状態 S_0 から状態 S_6 の到達可能性を調べる. これに対応する時相論理式は, $\neg EF(P_6)@S_0$ である. ここで, P_6 は, 状態 S_6 で成立する命題論理式の連言を表す. "@" の右辺は, 時相論理式が成立する状態 (可能世界) を表す. 図 1 で表現されるプロトコルで, 状態 S_0 から状態 S_6 への到達可能性は, $\neg EF(c_1 \wedge a_2)@S_0$ を調べる. $\neg EF(c_1 \wedge a_2) \equiv AG(\neg\ (c_1 \wedge a_2))$ であるから, この論理式は, 状態 S_0 から開始する全ての経路で $\neg(c_1 \wedge a_2)$ が成立することを表す. しかし, $(S_0,S_1,S_3,S_5,S_2,S_6)$ と (S_0,S_2,S_6) の遷移は, この条件を満たさない. 状態遷移図は可能世界モデルとはならないので, 状態 S_0 から S_6 への到達可能性が示される.

3.3 証明木の利用

モデルチェッカによる検証は,対象となる時相論理式に対する反例集合の要素になる時相論理式を追加することにより矛盾を導くことを基本としている.仕様に誤りがある場合、反例集合が生成される.モデルチェッカによる証明の過程で生成したこれらの時相論理式を,生成された順番に並べることにより証明木を作成することができる.証明木の一つの節点には,生成された一つの時相論理式が対応している.特に木の終端にある節点を葉節点,その他を内部節点と呼び区別する.内部節点は,高々有限個の枝を持つ.それらの枝には異なる節点が一つ対応する.証明木の根から一つの葉節点に至る系路上にある節点に対応する時相論理式の集合は,一つの反例集合を構成している.従って,証明木上の葉節点の個数は反例集合の個数に等しくなる.また,証明木の根に近い葉節点ほど早く生成された時相論理式に対応している.

プロトコルの仕様に誤りが確認された場合,その証明木と検証の対象となる時相論理式を調べることにより,誤りの原因の探索に利用できる.

例として,図 1 のプロトコルで状態 S_0 から開始する全ての系列が,再びその状態に戻ってくることを調べる.対象となる時相論理式は,$\neg AX(AF(a_1 \wedge a_2))@S_0$ である.この論理式で否定 $\neg$ を展開すると $\neg AX(AF(a_1 \wedge a_2)) \equiv EX(\neg AF(a_1 \wedge a_2)) \equiv EX(EG(\neg(a_1 \wedge a_2)))$ となる.この論理式に対して,モデルチェックを行うと 3 個の反例集合が生成される.この反例集合を基にして,図 2 に示す証明木を得ることができる.ただし,この図は一番最初に分岐する内部節点から下の部分木のみを表現している.

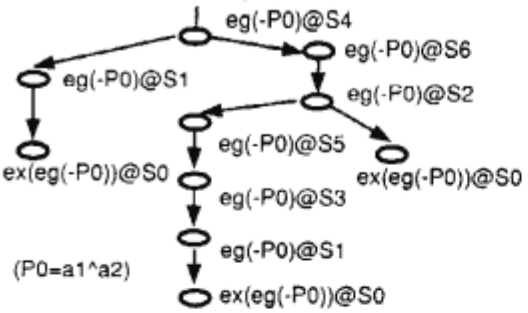

図 2: 証明木の例

この証明木は,3 個の葉節点を持っている.それらの葉節点は,検証の対象となる仕様を表現している時相論理式 $EX(EG(\neg(a_1 \wedge a_2)))@S_0$ が対応している.この例の時相論理式の場合,各葉節点から上に向かう系路上の節点で成立している時相論理式の状態を順番に並べると,状態 S_0 からの遷移系列を得ることができる.更に,この証明木の内部節点を調べると,各反例集合は $EG(\neg(a_1 \wedge a_2))$ なる時相論理式を含んでいることが分かる.各反例集合が共通に含んでいる状態は,S_4 である.$EG(\neg(a_1 \wedge a_2))@S$ の可能世界モデルによる解釈は,"状態 S から始まる遷移系列において,常に $\neg(a_1 \wedge a_2)$ が成立する系列の存在" を意味している.この場合,各反例集合は一つの遷移系列を含んでいる.以上の事から,各反例集合では,状態 S_4 へ至る系列において,$EG(\neg(a_1 \wedge a_2))$ が成立していることが分かる.遷移系列は,ループするか (次に遷移する状態が無いために) 終了するかのどちらかである.証明木から各系列はループを形成していないことが分かるので,このプロトコルは状態 S_4 でデッドロックを起こしていることが結論として得られる.

証明木の内部節点に対応する時相論理式には,検証の対象となる時相論理式の部分論理式が段階的に現われてくる.従って,それらの部分時相論理式が証明木の節点に出現する順序は,その時相論理式を段階的に分解することにより分かる.反例集合が複数ある場合も,それらの部分時相論理式が成立する状態が異なるだけで,同一の部分論理式を含んでいる.証明木の分岐は状態の遷移関係に,そして系路上の節点に対応する時相論理式が成立する状態を替えて続けて出現する回数は状態の条件に,各々依存する.以上の事から,証明木は時相論理式と状態の遷移関係の両方の構造を反映していることが分かる.

4 まとめ

本報告は,状態遷移図で表現されるプロトコルを対象とした検証方法を示した.検証方法では仕様の誤りが検出された際に,モデルチェックの過程で生成される反例集合から構成される証明木を解析することにより誤りの原因を追求できることを示した.ここで紹介したモデルチェッカ等は,並列定理証明器 MGTP でインプリメントされている [4].3.1 節で示した可能世界モデルによる解釈を利用することにより MGTP によるモデルチェッカの記述は容易にできる.これにより,処理の宣言的な記述を実現している.また,MGTP の定理証明器としての特徴から仕様誤り時の反例集合を収集する処理を特別に配慮する必要はない.

謝辞 本研究は第五世代コンピュータ・プロジェクトの一環として行われているものである.日頃御指導を頂く ICOT 研究部長代理長谷川隆三氏に深謝致します.

参考文献

[1] Emerson, E.A. et al.: *Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications*, ACM Trans. on Programming Languages and Systems, Vol.8, No.2, pp.244-263, 1986.

[2] Emerson, E.A. et al.: *Modalities for Model Checking: Branching Time Logic Strikes Back*, Science of Computer Programming 8, pp.275-306, 1987.

[3] Kripke, S.A.: *A Completeness Theorem in Modal Logic*, JSL Vol.24, pp1-14, 1959.

[4] Fujita, H., Hasegawa, R.: *A Model Generation Theorem Prover Using A Ramified-Stack Algorithm*, ICOT TR-606, 1990.