通信制御ソフトウェア開発における状態遷移表の実装効率化への取組み
～ZIPC を活用したモデル検査ツール開発への取組み～

富士通株式会社　共通開発本部　第一ソフトウェア開発統括部

松下　亮太郎

1. はじめに

我々が開発する製品は、携帯電話キャリア様向けの通信システムである。サービス制御や装置状態管理など、連続したプロトコル処理を用いるために様々な状態遷移表を活用してきた。

携帯電話サービスは半年ごとに機能拡張を続けており、開発線表も重複するため、１つの機能を複数の開発者が手を入れることも頻繁に発生する。

このため次のような問題が発生してきた。

- 過去の状態遷移表を拡張する際、既存の状態遷移への影響が把握しづらい
- 下流工程での状態遷移表の誤り検出は改修に工数がかかる

この状況から、上流工程における状態遷移表レベルでの品質確保を目的として、モデル検査技法の導入を検討してきた。

本稿では ZIPC の状態遷移表記述機能を利用することで、円滑に記述からモデル検査までを一連のツール化を実施した事例をご紹介する。

2. 通信ソフトウェア開発における状態遷移表の役割

移動体通信システムのソフトウェアでは、状態遷移表はソフトウェア構造を機能単位に分割した機能モジュール単位の設計生産物として作成される。主に２つの機能群に状態遷移表を利用している。

１つはサービス状態遷移の記述である。携帯電話から電話番号を入力して発信すると、端末や無線基地局と通信システム間の制御プロトコルが組み合わさってサ

ービスが提供される。端末ごとのサービス提供状態を状態遷移表で記述している。

もう一つは、装置運用状態遷移の記述である。様々な保守機能、運用機能動作を契機に装置状態が変化するため、状態遷移表を使って動作設計をおこなう。

これらの状態遷移表が、反復開発の中の機能追加で最大2000以上の状態に増加している。

3. モデル検査ツール SPIN モデルチェッカ

「モデル検査」はシステムの設計から導出されたモデルが形式仕様を満足するかどうかをアルゴリズム的に検証する手法である。我々は状態数の肥大化した通信システムにおける状態遷移表について、仕様策定・設計フェーズの検査の仕組みという点からモデル検査を選定した。

モデル検査にも多くのツールが存在するが、安定性や速度、完成度の高さから「SPIN」を選定することとした。

SPIN では、状態遷移系について、線形時相論理（LTL）による検査が可能である。線形時相論理による検査とは、言い換えれば1本の時間軸上で命題の真偽が単位時間ごとに変化する世界の検査である。

具体的には次のような検証が可能である。

a) **システム異常時のオフライン**

「異常ステータスになった後は、いつか必ずオフラインステータスになる」といった、時間経過の概念を含んだ検査が可能。

b) **複数プロセスの起動**

「管理プロセスからは、いつか必ず２つの通信プロセスが起動される」といった、非同期プロセスの状態組み合わせを検査可能。

c) **条件分岐による遷移**

状態遷移表における条件分岐表現において、それぞれの分岐先に遷移しているか検査可能。

d) **並行プロセスにおけるデッドロック**

人の目では追いにくい、非同期プロセスにおける状態の遷移を検査可能。

e) **通信異常時のメッセージ送信停止**

「通信異常時は、メッセージを送信しない」といった、状態を持つシーケンスの検査が可能。

4. モデル検査の作業の流れ

モデル検査は図１のような流れで検査を実施する。

設計書やインターフェース仕様書から状態遷移表を作成し、その状態遷移表で実現する動作モデルを SPIN が解釈できる Promela 言語へ変換する。同時に安全性と活性を確認する検査項目から線形時相論理式(LTL 式)を作成する。これらを SPIN にかけると、反例として得られる、条件を満たさなかった遷移経路を解析して状態遷移表を修正する。

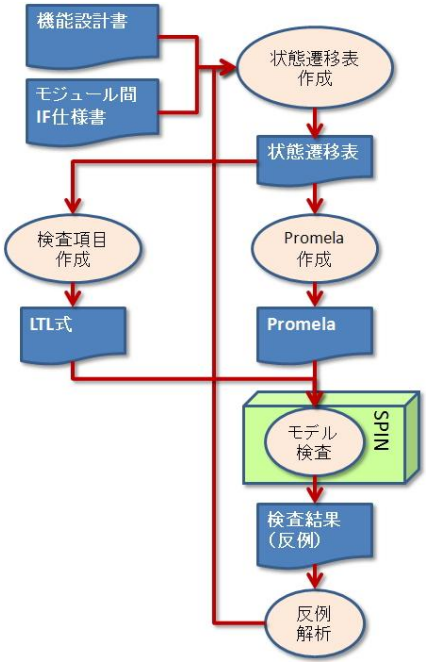

図１．モデル検査の流れ

5. モデル検査ツール導入への課題

SPIN 適用の課題を抽出するために、まずは SPIN の GUI ツールである XSpin をインストールし、実際のプロジェクトで作成された状態遷移表を借用して評価をおこなった。その結果、下記の２つの課題が抽出できた。

(1) Promela の自動生成

モデル検査をおこなうには状態遷移表から Promela を作成しなければならない。開発の担当者に Promela の言語仕様を習得させると、教育に手間がかかることや、Promela 言語を習得しても反例が検出された場合、Promela 言語のバグなのかモデルのバグなのか判断が難しいことから、この部分は自動化が必須である。

(2) 反例の解析の簡易化

モデルの複雑度に比例して反例も複雑となる。その上、SPIN の出力する反例ログは追跡が容易ではない。そこで、反例解析の補助を支援するツールが必要である。

これらの課題の解決を検討した。

6. Promela 生成ツールの開発

Promela 生成ツールを開発するため、まず実プロジェクトで作成している状態遷移表から特徴を抽出し、図２に示す３つの状態遷移表を検査対象とすることを決めた。

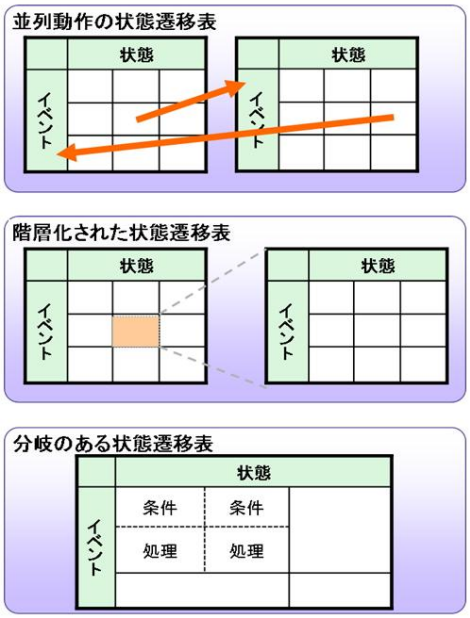

図２．ツールの対象検査モデル

これらの状態遷移表を対象に変換ツール実装に向け、更に２つの課題に直面した。

I. 状態遷移表の記述レベルの統一

実プロジェクトからサンプルの状態遷移表を集め、比較をおこなうと、状態遷移表の記述レベルがばらばらであること

が分かった。自動変換を実現するには記述レベルが統一されている必要がある。

状態遷移表の記述レベルを統一するために、当初Microsoft Office Excelによる独自フォーマットの開発を試行したが、記法の統一までには至らなかった。そこで、富士通グループでも採用実績のあるキャッツ社のツール「ZIPC」に着目した。

ZIPC を利用することで、上記３パターンの状態遷移表が記述できることを確認した上で、さらに記述上の誤りをツール上で検査することができることが分かったため、状態遷移表の記述には ZIPC を採用することにした。

キャッツ社に依頼し、ZIPC の STM フォーマットを開示していただき、XML で記述された状態遷移表を Promela に変換するツールを独自に開発することにした。

II. 階層化状態遷移表の Promela 変換

Promelaの自動生成ツールの設計を進める中で、階層化状態遷移表（図３）の変換アルゴリズムが複雑化するため、実装を簡素化する手段を検討した。

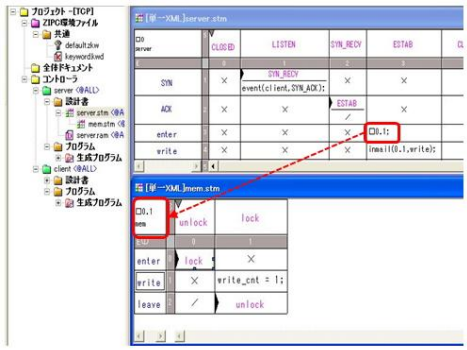

図３．階層化した状態遷移表

例えば、次のような実装方法を用いている。

例１）親と子の状態遷移表は、それぞれが非同期で動作するわけではないので、Promelaのプロセスを親と子で分割せず、子は親の状態遷移表にプログラム内部で展開しひとつの大きい状態として扱う。

例２）子の状態遷移表への遷移の表現には「□0.1」や「inmail(0.1,イベント)」といった表現がある。「inmail(0.1,イベント)」と記載された場合は、指定されたイベントが選択されるが、「□0.1」と記載された場合は、イベントが指定されていない。よって子の状態遷移表にイベントが発生する場合、子の状態遷移表のイベントを非決定的に選択する。

以上のような工夫をおこなうことで、約２か月で自動生成ツールを完成することができた。

7. 反例解析支援ツールの開発

反例解析ツールでは、SPIN が出力する反例ログを表１の項目で整理して、csv ファイルに出力する。

表示名	表示和名	説明
SequenceNumber	シーケンス番号	状態遷移系での流れを表す番号
ProcessNumber	プロセス番号	プロセス（関数）を表す番号 タスク１つに対して １つの番号が１番から振られます
StatusVal	ステータス番号	状態遷移系での状態番号
XXX_st (例: client_st)	タスクステータス	その時点における状態遷移表のステータスの番号 状態遷移表において左から0番になります （XXXは状態遷移表名）
XXX_ev (例: client_ev)	タスクイベント	その時点における状態遷移表のイベントの番号 状態遷移表において上から0番になります （XXXは状態遷移表名）
任意の表示名 (例: localVal)	定義変数	状態遷移表で定義した変数の値
LOOP	ループ表示	ループ開始からループ終了までのループ状態になった旨を示す文字出力項目 ループ開始: LOOP START ループ中 : LOOP ループ終了: LOOP END

表１．反例解析結果項目定義

すると、図４のように情報が整理され、変数や状態変化が把握し易くすることが

できた。

	A	B	C	D	E	F	G
1	SequenceNumber	ProcessNumber	StatusVal	client_st	client_ev	server_st	server_ev
2	6	2	1				
3	19	2	4	0			
4	20	2	6	0	0		
5	25	1	3	0	0	0	
6	26	1	5	0	0	0	0
7	31	2	38	1	0	0	0
8	32	2	53	1	3	0	0

図４．反例解析結果例

8. モデル検査 Web システム

モデル検査を開発プロセスに導入する仕組みとして、モデル検査全体を一括して管理できる「モデル検査 Web システム」を構築した。（図５、図６）

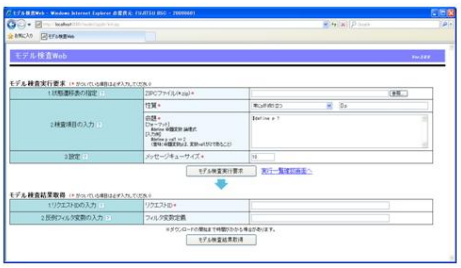

図５．モデル検査 Web システム

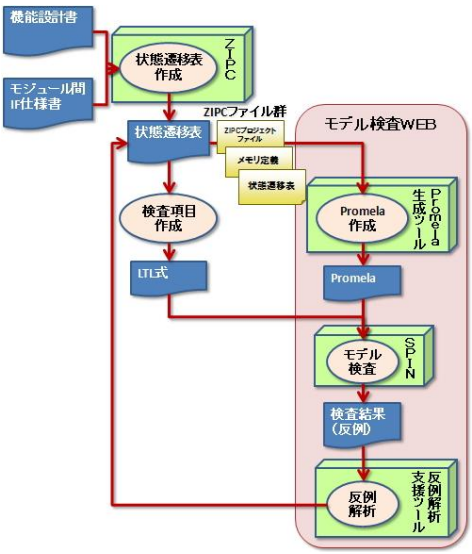

図６．モデル検査 Web システムの構成

モデル検査 Web システムでは、一画面でモデル検査に必要な一連の処理をおこなうことが可能である。また、検査項目である LTL 式の記述に選択肢形式と直接入力を併用することとした。この選択肢のおかげで、時相論理の教育も最小限にとどめることができる。

今年度の開発から、モデル検査 Web システムを実プロジェクトにおいて導入を開始した。

9. 現状の課題と今後の展開

利用した開発者にアンケートをおこなった結果、ツールには次の課題がある。

1. **検査項目抽出ガイドラインの作成**
   検査項目から時相論理（LTL 式）への変換は難易度が高いため、ガイドラインが必要
2. **LTL 式の妥当性検査**
   検査項目から変換後の LTL 式の文法チェックが必要
3. **LTL 式の一括投入機能の追加**
   LTL 式は現在のモデル検査 Web システムでは１つずつしか投入できない。検査項目は複数あるので、一括投入の仕組みが必要

今後はこれらの課題を解決するツールの改修や整備をおこないつつ、さらに適用プロジェクトを増やすために展開活動を推進する。

参考資料

［１］産業技術総合研究所システム検証研究センター　著、モデル検査[初級編]

［２］産業技術総合研究所システム検証研究センター　著、モデル検査[上級編]
［３］Mordechai Ben-Ari　著　中島震 監訳 谷津弘一・野中哲・足立太郎　共訳、SPIN モデル検査入門
［４］富士ソフト株式会社　小池隆、ZIPC WATCHERS Vol.13 モデル検査支援ツールと ZIPC との連携による設計懸賞の導入事例