3K-04

ペトリネット検証ツール HiPS における LTL 仕様に基づいた On-the-fly/Fluent モデル検査

張江　洋次朗 †　　和崎　克己 ‡

† 信州大学理工学系研究科情報工学専攻　　‡ 信州大学工学部情報工学科

1　はじめに

本稿では，ペトリネットより生成される状態空間を対象に，LTL 式による on-the-fly モデル検査器の設計について記す．筆者らの研究グループによって開発・公開されているペトリネット設計・検証ツール HiPS がある．HiPS は，ペトリネットの初期マーキングを根とし発火系列によって構成される状態空間を生成する機能が備わっている．HiPS は外部ツールと連携することで，状態空間に対するモデル検査を行うことができる．現状ではモデル検査を適用するためには全ての状態空間の生成を行わなくてはならず，時間的制約から大規模モデルへの適用は難しい．モデル検査の効率化として，on-the-fly 手法を用いたモデル検査器の HiPS ツールへの組み込みを行った．

2　ペトリネット検証ツール HIPS

離散事象を視覚的，数学的に記述するツールとしてペトリネットがある．ペトリネットは並行性，非同期性，並列性，非決定的選択といった性質・動作をもつモデルの表現が可能である．初期マーキングから発火可能なトランジションの発火により新しいマーキングを得る．このプロセスはマーキングをノード，トランジションをエッジとする木表現に帰着する．特に，ネットが有界であるとき可達木と呼び，全て異なったマーキングをもつノード集合による有向グラフとして表現した場合，可達グラフと呼ぶ．可達グラフをモデルの状態空間として考えることができる．ペトリネット設計ツールとして筆者らの研究グループによって開発・公開されている HiPS(Hierarchical Petrinet Simulator) がある．直感的で一般的な操作方法の GUI とペトリネットの階層化と時間ペトリネットに対応し，作成したペトリネットモデルに対してランダムウォークシミュレートし挙動を観察することが可能である．HiPS ではシステム全体の網羅的な振る舞いを初期状態から到達可能な全状態集合である状態空間として生成する [1]．状態空間はラベル付き遷移系 (Labelled Transition System:LTS) で記述され，LTS のファイルフォーマットである Aldebaran-Automaton 形式で出力される．状態空間に対する網羅的な探査を行うことで，システムが要求する性質を満たしているかを検証することができる．オートマトンを用いたモデル検査を対象とするため，初期状態から得られる連結グラフであるような状態空間を必要とする．そのため，シングルスレッドによる動作で状態空間の生成を行う．状態空間生成器から得るトレースと観察したいパターンとともに外部ツールを用いることで，トランジション発火列に関するモデル検査を実行する．

†Yojiro HARIE　‡Katsumi WASAKI
†Graduate School of Science and Technology, Shinshu University
‡Department of Information Engineering, Faculty of Engineering, Shinshu University

3　モデル検査器ユーザインターフェース作成

モデル検査を HiPS ツール上で実現するために，状態空間生成だけでなく，検査する性質の記述をツール上で行えることが望ましい．このフォーム上には，LTL で用いられる作用素 ([](Global)，<>(Future)，X(Next)，U(Until) などの時間演算子と，!，∨，∧ などの古典的な論理演算子) を示すボタンを配置した．ペトリネットのトランジションを検査式の論理命題として表現するが，大規模ネットを検査対象とする場合，ペトリネット内のインスタンス数が膨大になるため，ネットを参照して検査式を記述することは困難である．記述支援として，検査対象トランジションの LTL 記述フォームへの自動表示機能を付与した．キャンバス上の各要素にデザインエントリとしての情報 “proposition name” を追加した．proposition name 欄に任意の名前を記述がある場合，対象トランジションと認識し，フォーム上へ表示される．

4　on-the-fly LTL/Fluent モデル検査器

モデル検査とは，システムを記述したモデルが要求する性質を満たすことを自動的に検証する技術である．システムが要求する性質の記述に時相論理を用いて，線形時相論理 (Lenear Temporal Logic: LTL) で形式的に表現する．ここでは，オートマトンに基づく LTL についてのモデル検査法を前提とする．モデル検査の効率化手法のひとつに on-the-fly 検証がある．on-the-fly 検証とは，状態空間の構築と並列して探索を行うことで，受

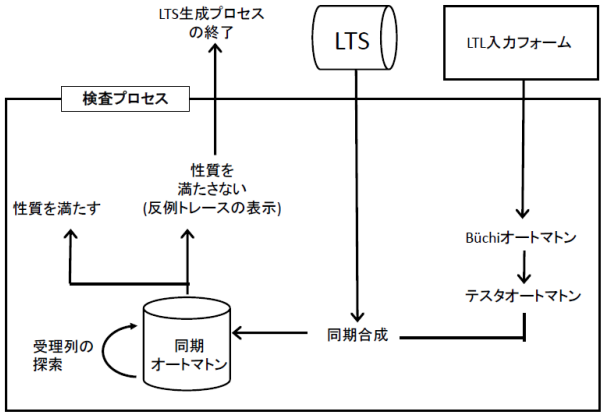

図 1: 検査プロセスの流れ

理列を発見した場合に，すべての状態空間の構築より前に探索および構築を終了させる方法である．満たすべき性質を記述した LTL 式より Büchi オートマトンを構成し，システムオートマトンとの同期積をとる．同期積オートマトンが空でない場合，受理列が具体的な反例となる．ペトリネットの状態空間は LTS で表現されるため，同期積を行うためには仕様オートマトンも LTS の形式でなくてはならない．Büchi オートマトンを受理状態をもつ LTS へ変換する方法がある [2]．図 1 に作成したモデル検査の内部処理の概略を示す．記述フォーム上より得た LTL 式を Büchi オートマトンへ変換し，さらに受理状態を持つ LTS であるテスタオートマトンへ変換する．テスタオートマトンとシステムオートマトンを同期合成を行い，受理列の探査によるシステム検証を行う．

5 実装と考察

on-the-fly モデル検査の実装に際して，既に HiPS 上に実装されている状態空間の生成部との並列処理を行う．モデル検査プロセスと既に実装済みである状態空間生成プロセス間で，IPC チャンネルを用いたプロセス間通信によって on-the-fly 検査を実現した．検査プロセスはキューから LTS を pop し，逐次的にシステムオートマトンとの同期合成を行う．終了フラグの初期値は false に設定しており，生成終了時あるいは検査終了時には終了フラグを true へ値を変更する．終了フラグが false かつキューが空であるならば，検査プロセスは条件を抜けるまで待機状態を遷移し続ける．検査プロセスにて反例を検知した場合には，終了フラグを true に変更し，生成部は終了フラグの値の変更を確認時に生成を終了させる．

検査対象として，ペトリネットによる在庫をひとつ

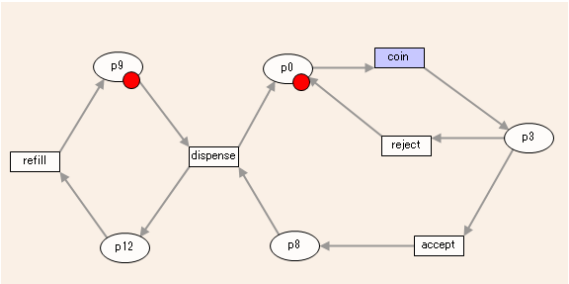

図 2: 在庫付き自動販売機のペトリネットモデル

とする図 2 のような自動販売機モデルを対象とした．coin を挿入し，受理できないような coin である場合は reject を行い初期状態へ遷移する．coin を受理した場合には，在庫を消費して dispense を行う．在庫の補充には refill によって行う．補充に関するプロパティである $\neg([](accept \rightarrow <> refill))$ を検査式として与える．検査結果として，反例のひとつである coin→accept→dispense →coin→reject→coin→···(coin→reject でループしている) が得られた．この結果は在庫に関して有限なモデルであるが，不正 coin に対する操作に関して制限してないことに起因していることが分かる．このモデルに対しては，coin の不正に関しては有限回という制限を与えたモデル修正が考えられる．

6 まとめ

HiPS は外部ツールを用いることでモデル検査を行っていたが，物理的・時間的問題から大規模なモデルの検査が困難であった．on-the-fly モデル検査器を導入によって逐次的な検査を行えるようになった．今後の課題として，仕様記述の簡略化とモデル修正のサポート, 実行時間やメモリ使用量といった検査器の性能検証が挙げられる．代表的な検証性質をマクロ化した SpecPattern の導入を行いたい．ネットの初期マーキングからトランジションの発火によるトークンの遷移として反例トレースを GUI 上でアニメーションのように確認できる，ガイド付き反例トレース実行機能の追加を行いたい．得られた受理状態までの発火系列を初期状態から順次発火させることで，HiPS ツール上で反例トレースの視覚的な確認ができる．任意の機会による段階的な表示を可能にすることでモデル修正のサポートを行いたい．

参考文献

[1] 太田淳也，和崎克己，ペトリネット援用ツールを用いたモデル設計とポスト検証ツール向け状態空間生成アルゴリズム，第 12 回情報科学技術フォーラム (FIT2013)，pp.171-174，2013.

[2] D. Giannakopoulou and J. Magee, Fluent model checking for event-based systems, Proceedings of the 11th ACM SIGSOFT Symposium on Foundations of Software Engineering 2003 Conference, ESEC/FSE 2003, September 1-5, pp.257-266, ACM, 2003.