A-015

ペトリネット設計検証ツール HiPS における On-the-fly LTL モデル検査器

On-the-fly LTL Model Checker on the Petri Net Design Tool : HiPS

張江 洋次朗 †　和崎 克己 ††
Yojiiro Harie　Katsumi Wasaki

1　はじめに

離散事象システムのモデリングに有用なツールとしてペトリネット [1] がある．HiPS(Hierarchical Petri net Simulator) は，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 [2]．ペトリネットが表している遷移関係・状態値を元に，モデルの動的な性質の検証を効率よく行うことは重要である．システムの信頼性の検証において，テスト・シミュレーションを用いた方法があるが，いずれもシステム内の不具合が存在しないことの証明にはならない．数学的解析を用いた自動的な検査法としてモデル検査がある．モデル検査はバグ解消の有益な手段のひとつであるが，複雑なシステムを対象とする場合，状態爆発を起こす可能性がある．

モデル検査手法の効率化として，on-the-fly 実行がある．モデル検査における on-the-fly 実行とは，状態空間生成と並列してモデル検査を行うことである．本研究では，線形時相論理（LTL）で記述した仕様を満足するか検証を行う，オートマトンベースのモデル検査器の HiPS ツールへの組み込みを行った．既研究により HiPS に実装済である状態空間生成器によって，ペトリネットの初期マーキングから出発したトランジション発火系列が得られる．システムが満たすべき性質の否定をとった LTL 式から Büchi オートマトンを生成し，逐次的に得られた発火系列に対する監視を行う．受理された発火系列は反例トレースであり，結果は HiPS ツールによって可視化される．実装したモデル検査器は on-the-fly 実行することで，検査終了までの実行時間の短縮を図った．

2　従来研究

2.1　ペトリネット

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．ペトリネットは，トランジションとプレースというノードからなる 2 部グラフであり，トランジションとプレースを結ぶアークとトークンを加えた 4 つの構成要素からなる．ペトリネットは，プレースを事前・事後条件，トークンを条件の成立，トランジションを事象と解釈ができる．ネットの初期状態を表す初期マーキングから，トランジションの発火による状態の変化によりシステムの動的な挙動が記述できる．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，すべて異なったラベル (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．

2.2　ペトリネット設計検証ツール HiPS

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 Petri net Simulator) がある [2]．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 [3]．図 1 に HiPS の操作画面と Muller’s C-element 回路のペトリネットモデルを示す．ペトリネットの性質には，初期マーキングに依存する動的性質と，ネットの構造に関する構造的性質があり，それらのペトリネットの性質に基づく解析機能が実装されている．可達グラフ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行える．

2.3　状態空間生成

ペトリネットにおける状態空間は有界な可達グラフとして与えられる．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すべての状態の集合である状態空間として生成する．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 [4] で出力される．LTS は状態間の遷移にラベル付けがされており，事象に基づくシステムの振る舞いを記述している．

C#で実装されている HiPS のコア部分に加えて，並列化ライブラリや実行速度の観点から，状態空間生成については C++ で実装されている．状態空間生成の並列化にも対応しており，C++ 向け並列化ライブラリである Intel TBB を使用し，最適化コンパイラとして Intel C++ Compiler を使用している．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．

2.4　HiPS におけるモデル検査器 UI 作成

モデル検査を HiPS ツール上で実現するために，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を付与し，LTL 式の命題として利用するためのユーザイン

† 信州大学大学院理工学系研究科，Graduate School of Science and Technology，Shinshu University.
†† 信州大学工学部，Faculty of Engineering，Shinshu University.

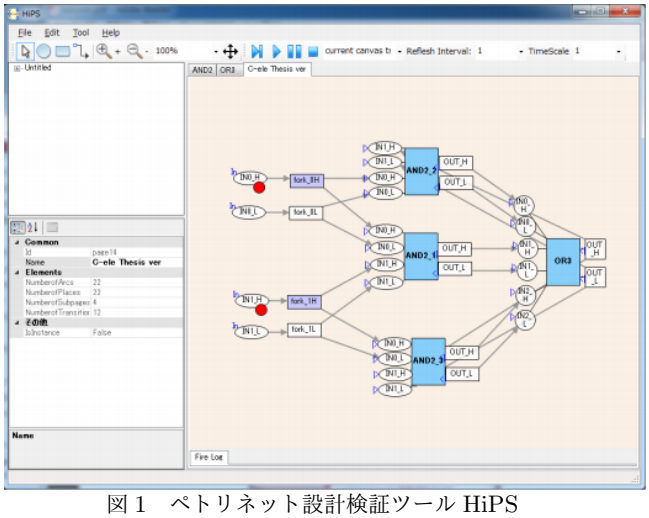

図 1　ペトリネット設計検証ツール HiPS

ターフェースを作成した．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．モデル規模が増大すると，ペトリネット内のインスタンス数が増え，監視対象のトランジションがイベントとして何を意味しているのかを直感的に理解することが困難になる．そのため，事象に関する性質を記述する際に，設計したモデルの要素 id を参照する方法では，LTL 式入力の記述コストが高くなるという問題がある．

そこで，サブページの要素として監視対象であるかを規定する項目“PropositionName”を付与した．この項目にネット上から入力があった場合，監視対象となるトランジションの id 等のデータをフォーム上に表示する．表示されたトランジション毎に入力ボタンが付随しており，フォーム上の LTL 演算子ボタンと併せて記述に利用することで，LTL 式の記述が容易に行える．図 2 に HiPS に実装した LTL 入力フォーム画面を示す．

3　LTL モデル検査

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 (LTL) や計算木論理 (Computation Tree Logic: CTL，CTL) で形式的に表現する．検証性質として，決してある状態・遷移が起こらないことを意味する安全性と，望ましいことがいつか起こることを意味する活性がある．本稿では，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．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．

3.1　LTS におけるモデル検査

LTS はシステムで観察される事象の順序を記述するモデルなので，事象に関する性質を LTL で記述できることが望ましい [5]．このような性質を記述するために，事象によって真理値が定義される述語である流動の概念と，その流動の真偽について推論する LTL である流動 LTL（FLTL）が提案されている [6]．Büchi オート

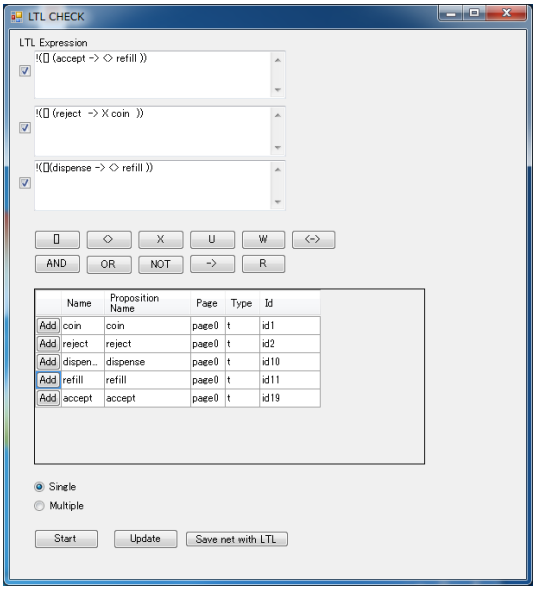

図 2　LTL 入力フォーム画面

マトンの各遷移は，モデルに現れる命題によってラベル付けされている．一方，FLTL で扱う命題は流動であるが，流動は LTS 上には明示的にラベル付けされていない．よって，FLTL 式の Büchi オートマトンを構成しても，両者のラベルの集合は一般に異なるため，モデルとの積オートマトンが受理する軌跡を探索することによる検証ができない．そこで，Büchi オートマトンを受理状態をもつ LTS へ変換する方法が知られている．変換した LTS の遷移に関するラベルは対象のモデルに現れる事象で記述されている．

3.2　on-the-fly モデル検査

モデルの規模が大きくなると，状態数が爆発的に増え処理時間が膨大になるという問題がある．この問題に対する状態空間探索の効率的な手法として，on-the-fly 法 [7] がある．on-the-fly 法とは，状態空間の構築と並列して探索を行うことで，受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．モデル検査は前述の状態爆発という問題を抱えており，メモリ・実行時間の削減について on-the-fly 実行は有用な手段のひとつである．

3.3　Nested Depth First Search

Nested Depth First Search(Nested DFS) [8] は，LTL 検査において逐次最適な探索アルゴリズムである．Büchi のオートマトンの受理条件が，受理状態の無限回通過であり，Nested DFS では二重に DFS を用いている．はじめの DFS で，初期状態から到達可能な受理状態の探索を行う．受理状態を発見した場合，二番目の DFS でその受理状態から走査をはじめ，閉路の探索を行う．Nested DFS は，受理サイクルが存在する場合，全状態空間を構築する前に探索が終了する，on-the-fly level2 の性能を有する [9]．

4　実装

4.1　LTL から Büchi オートマトンへの変換

LTL は性質について直観的な記述が可能だが，LTL そのものを利用した検証が難しい場合がある．そのため，式と等価なオートマトンへ変換を行い，得られたオートマトンを用いるという検証手法が知られている．LTL から等価な NeverClaim への変換を行うツールとして LTL2BA [10] を使用した．本研究では，C で記述されている LTL2BA を，C++ として書き換え，HiPS への組み込みを行った．LTL2BA では受理状態や状態間の遷移構造を双方向リストで，遷移については遷移名を保存したテーブルを参照することで Büchi オートマトンを表現している．HiPS への実装のために，Büchi オートマトンとして得られる構造と遷移名についてひとつまとめて，C++ のコンテナクラスである vector を用いた新しい構造への変換を行った．入力として満たすべき性質を表現した論理式の否定を与える (図 3)．

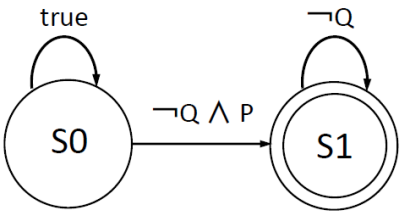

図 3　BA(¬ ([](P-><>Q))) への変換例

4.2　Büchi オートマトンから LTS への変換

流動に関して記述された Büchi オートマトンをそのまま LTS におけるモデル検査に用いることはできない．入力した式について LTL2BA より得られた構造に対して，受理状態をもつ LTS(テスタオートマトン) への変換を行う．テスタオートマトンはアルファベットが対象モデルの事象の有限集合であるような Büchi オートマトンであり，変換前と同様の受理条件をもつ．LTS で記述されたシステムオートマトンと，テスタオートマトンとの LTS 同士の並列合成により積オートマトンを求める．

4.3　受理状態の探査

性質オートマトンとシステムオートマトンとの同期積オートマトンを生成し，Nested DFS アルゴリズムを用いて積オートマトンに対する受理状態の探査を行う．受理状態がない場合，検査の終了と同時にシステムが性質を満たしているという通知を UI 上へ表示する．受理状態が検知された場合，初期状態から受理状態へ至るまでの反例トレースの表示，状態空間生成プロセスの停止を行う．図 4 は LTL 式入力から検査終了までの検査プロセスの内部動作を図式化したものである．

4.4　状態空間生成部とのプロセス間通信

on-the-fly モデル検査の実装に際して，すでに 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状態空間生成プロセス間で，LTS を格納するスレッドセーフなキュー構造と，終了条件となる終了フラグを共有メモリとするプロセス間通信を行う (図 5)．検査プロセスを親プロセスとして，

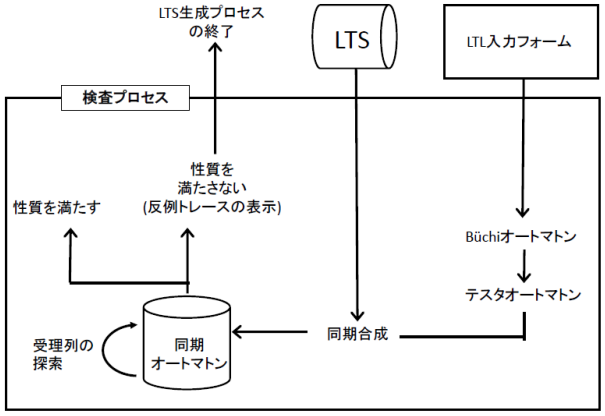

図 4　検査プロセス内部

状態空間生成プロセスをフォークする．生成プロセスは LTS の形式で状態空間を共有メモリ上のキューに push し，検査プロセスはキューから pop し LTS をとりだし逐次的にモデル検査を行う．もし，終了フラグがたっていない，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．状態空間生成プロセスは，すべての状態空間生成が完了したのならば，LTS を共有メモリ上のキューに push したのちに終了フラグをたてる．検査プロセスにて，すべての状態空間生成の前に反例を発見した場合は，生成プロセスに検査終了を通知する終了フラグをたてる．生成プロセスは終了フラグを確認すると，状態空間生成を停止する．

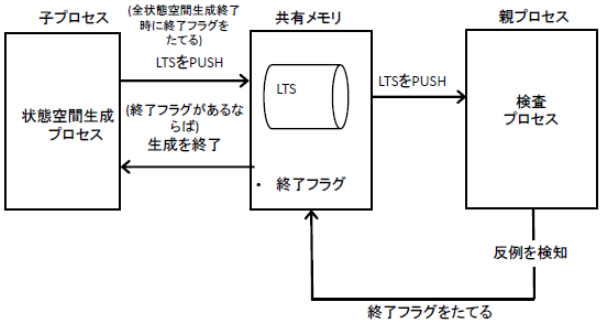

図 5　検査プロセスと状態空間生成プロセスとのプロセス間通信

5　課題

5.1　反例のガイド付実行

モデル修正のための反例分析に，反例の可視化を行うことは重要である．LTL 式を対象とするモデル検査器 SPIN の GUI サポートツール iSpin では，反例トレースの可視化機能が備わっている．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の実行機能を HiPS に追加したい．これは，HiPS の既存機能であるランダムシミュレーション機能の，特定トランジションに対する実行の他ならない．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

反例トレースの確認を視覚的に行えるようにする．反例トレースの自動的な表示だけでなく，利用者の任意のタイミングによる段階的な表示を可能にすることで，より詳細な反例分析のサポートが期待できる．

5.2 SPEC PATTERN

時相論理は直観的に検証性質の記述ができるが，範囲の指定といった複数の条件を付与した式は複雑になりえる．検証性質にはいくつかの典型的なパターンがあり，頻繁に用いられる式のリポジトリが行われている．代表的な検証性質をマクロ化した，SpecPattern [11] が考案されており，HiPS への導入を行いたい．本研究で作成した LTL 入力フォームと同様にネットで選択したトランジションをフォーム上に表示させる．表示したトランジションを命題として検証性質パターンにあてはめ LTL の記述を行う．

5.3 状態空間縮約

状態空間縮約を行うことで，状態空間生成の効率化が図れる．SPIN などのツールでは partial order reduction 法による縮約を行っているが，性質の記述の際に LTL で用いられる Next 作用素を使えないという制約がつく．HiPS では，状態空間生成部と検査部を別プロセスで行っており，生成部でのモデル縮約化を適用させることが望ましい．既存アルゴリズムとの比較や生成プロセスを考慮し，HiPS における効率的な縮約アルゴリズムを導入したい．

6 まとめ

HiPS 上で LTL モデル検査を行えるようにし，on-the-fly 実行を導入することで，処理時間，メモリ使用量の削減を図った．モデルの設計から検査まで GUI 上ですべて行えるようにすることで，操作性の向上を目指した．HiPS 上でペトリネットによるモデルを設計し，そのモデルに対する検査を行い評価実験を行っていく．

参考文献

[1] T. Murata : “Petri Nets : Properties,Analysis and Applications” ; Proc. of IEEE, 77(4), 1989.
[2] HiPS Tools : http://sourceforge.net/projects/hips-tools/.
[3] 太田, 和崎 : “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” ; FIT2013（第 12 回情報科学技術フォーラム）講演論文集, (A-015), 171-174, 2013.
[4] CADP Toolbox, VASY/INRIA : http://cadp.inria.fr/.
[5] 熊澤 : “高信頼性ソフトウェアシステムの実現のためのモデル修正技術に関する研究” ; 博士論文, 東京大学総合文化研究科, 2011.
[6] D. Giannakopoulou, J. Magee : “Fluent model checking for event-based systems” ; pp.257266, 2003.
[7] E. M. Clarke, O. Grumberg and D. A. Peled : “Model Checking” ; MIT Press, 1999.
[8] C. Courcoubetis, M. Y. Vardi, P. Wolper and M. Yannakakis : “Memory Efficiend Algorithm for the Verification of Temporal Properties” ; LNCS 531, pp.233-242, 1990.
[9] 安田, 吉田, 上田 : “LMNtal 並列モデル検査における状態生成数削減及び高速化” ; The 27th Annual Conference of the Japanese Society for Artificial Intelligence, 2013.
[10] P. Gastin, LTL2BA : http://www.lsv.ens-cachan.fr/ gastin/ltl2ba/, 2014-11-28.
[11] SAnToS laboratory, SpecPatterns : http://patterns.projects.cis.ksu.edu/, 2015-6-16.