5L-01

リアクティブシステム仕様の分割実現可能性判定法

根本　陽菜†　　島川　昌也†
拓殖大学工学部情報工学科†

1. はじめに

近年の情報システムの複雑化に伴い，場当たり的なテストやレビューによる欠陥の発見が困難になってきている．そこでシステムの安全性の保証や欠陥の発見を行う形式検証への期待が高まっている．形式検証では，人力では見つけにくい欠陥を計算機によって自動で検出することができるが，計算コストの高さが実用上の障壁となっている．特に「実現可能性（環境がどのような要求をどのような順序で生起しても，仕様を満たすような応答を過去の要求履歴から決定できる実現システムが存在すること）」と呼ばれる性質の判定は，高い計算コストを伴うため，適用できる仕様の規模が限られている．そこで本研究では，構成したωオートマトンに局所情報の捨象を適用する分割検証法を提案する．複数のモジュールに分割した仕様から無限木オートマトンを構成し，分割先でのみ現れる局所情報の捨象を適用したものを統合，分析する手法を与えることで計算コストの削減を目指す．

2. リアクティブシステムの検証

本研究では，「リアクティブシステム」と呼ばれる，環境とのインタラクション維持を目的とするシステムのふるまい仕様を検証の対象とする．例えば，エレベータの制御システムや原子力発電所の制御システムが挙げられるが，これらは問題が生じた場合のリスクが大きいため，安全性が強く求められる．線形時間論理で記述されたエレベータの制御システム仕様を例に挙げる（図1）．一見，何の問題もないように思えるがこの仕様には欠陥が存在する．「開ボタンと閉ボタンが同時に押された後，閉ボタンが押され続けた場合に，1 と 2 の両方を満たすシステムが実現できない」からである．よってこの仕様は「実現不能である」と判定される．リアクティブシステムは，入出力のタイミングが重要であるため，線形時間論理などでシステムのふるまいに関する仕様を厳密に記述し，検証することが有効である．

3. 提案手法

通常の実現可能性判定（一括検証）は，以下のような手順で行う（図 2）．

① 仕様から，それと等価な非決定性ω木オートマトン A を構成
② 構成したオートマトン A を分析

ここで構成されるオートマトン A のサイズは膨大となるため，構成・検証のコストは極めて高くなる．

1. 常に，開ボタンが押されたら，いつかドアを開く
   $\boldsymbol{G}$ (open_btn → $\boldsymbol{F}$ open);
2. 常に，閉ボタンが押されたら，ドアを閉じる
   $\boldsymbol{G}$ (close_btn → ¬open);
3. 常に，開いているときに開延長ボタンが押されたら，閉ボタンが押されるまでドアを開けておく．
   $\boldsymbol{G}$ ((open_ex_btn ∧ open) → ((open $\boldsymbol{U}$ close_btn) ∨ ($\boldsymbol{G}$ open)));

図 1. エレベータの制御システム仕様

分割検証は，一括検証の計算コスト問題を緩和させることを目的とし，以下のような手順で行う（図 2）．

① 仕様をいくつかのサブモジュールに分割
② 各サブモジュール仕様について，それと等価なオートマトン A_i を構成
③ 構成したωオートマトン A_i に部分手続きを適用
④ 適用後のωオートマトン A_i' を統合
⑤ 統合したωオートマトン A' を分析

ステップ③で適用した部分手続きでは，各サブモジュールにおける局所的な情報の除去を行う．既存手法[1]では，受理不能状態の除去など単なるオートマトンの縮小化（受理言語が変化しない変換）が適用されているが，本研究の手法では，各サブモジュールにおける局所的な情報の除去（受理言語が変化する変換）も適用する．このアイディアは，文献[2]の充足可能性判定の分割検証法において適用されている処理を参考に，本手法で実現可能性判定の効率向上に向けてアプローチしたものである．分割したサブモジュール i にのみ現れるローカルな応答イベントは，統合後の検証で不必要な情報であるため，捨象してもオートマトンに問題はなく，簡約により実現可能性判定の効率を大幅に引き上げることが期待できる．

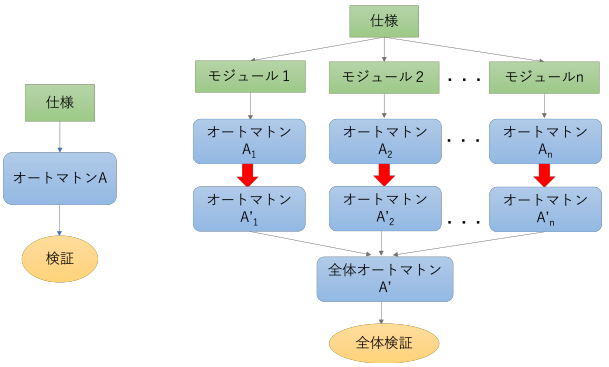

図 2. 一括検証と分割検証

Towards compositional realizability checking for reactive system specifications
†Nemoto Haruna and Shimakawa Masaya
Takushoku University

4. 実験

本研究では，提案手法を Python と BDD(Binary Decision Diagram)を用いて実装した．オートマトンの遷移条件の処理に BDD を利用し，その他の部分については Python 組み込みのデータ構造を利用した．

その実装をもとに，提案手法の効率を調査する実験を行った．具体的には，(i) 一括検証と提案手法の判定時間の比較，(ii) 既存の分割検証も可能な実現可能性判定ツール Acacia+[3]と提案手法の判定時間の比較を行った．ベンチマークには，The Reactive Synthesis Competition[4]から n ビットのラッチ回路仕様（n_ary latch），n ビットのシフト演算回路仕様(shift)の 2 つの仕様を使用した．分割検証では，それぞれ n 分割，2 分割，3 分割での判定時間を測定する．

一括検証と提案手法の比較結果を表 1 と表 2 に示す．実行時間が 3600 秒を超えた場合はタイムアウトとし，TO と表記する．一括検証と提案手法による実行時間を比較した結果，提案手法による実現可能性判定の大幅な効率向上が確認できた．

仕様の分割数による実行時間を比較すると，分割数によって処理効率に差が見られた． n ビットのラッチ回路仕様では，n 分割が最も速く，2 分割が最も遅くなった．一般に，仕様の分割検証ではステップ④のオートマトンの統合のコストが最も大きくなる．しかし，今回使用したラッチ回路の仕様では，提案手法によるオートマトンの簡約の効果が大きく現れたために，オートマトンの統合にかかるコストが，ステップ②の各サブモジュールのオートマトン A_i の構成にかかるコストを大幅に下回るほど削減された．そのため，A_i の構成のコストが低い n 分割での効率が良かったのだと考える．次に，n ビットのシフト演算回路仕様では，2 分割と 3 分割による処理が比較的速く，n 分割が最も遅くなった．これは分割数が大きいほど分割・統合のオーバーヘッドが大きくなるためであると考える．

表 1. 一括検証と提案手法の比較（n_ary latch）

	一括	提案手法		
		n分割	2 分割	3 分割
n=6	11.62s	0.08s	0.11s	0.07s
n=8	266.90s	0.10s	0.42s	0.12s
n=10	TO	0.16s	2.76s	0.32s
n=12	TO	0.38s	37.94s	0.72s
n=14	TO	1.27s	254.18s	4.42s
n=16	TO	4.18s	1437.01s	10.69s
n=18	TO	19.98s	TO	31.09s

表 2. 一括検証と提案手法の比較（shift）

	一括	提案手法		
		n分割	2 分割	3 分割
n=10	0.75s	0.13s	0.09s	0.09s
n=12	1.44s	0.40s	0.07s	0.11s
n=14	64.54s	1.03s	0.55s	0.29s
n=16	655.80s	4.58s	2.52s	2.26s
n=18	1412.43s	24.02s	11.37s	11.24s
n=20	1853.69s	123.54s	61.60s	60.30s
n=22	3413.25s	520.35s	274.33s	278.37s

既存ツール Acacia+と提案手法による実行時間の比較結果を表 3 と表 4 に示す．こちらも提案手法による実現可能性判定の効率向上が確認できた．Acacia+と今回の提案手法の実装では，利用するデータ構造が異なるなど実装方針に違いはあるものの，オートマトンを縮小化する処理に加えて提案したローカルイベント情報の捨象を行う処理が効いたためであると考える．

表 3. 既存ツールと提案手法の比較（n_ary latch）

	Acacia+			提案手法		
	n分割	2 分割	3 分割	n分割	2 分割	3 分割
n=6	0.03s	0.20s	0.02s	0.08s	0.13s	0.07s
n=8	0.06s	0.41s	0.13s	0.10s	0.32s	0.12s
n=10	0.16s	2.03s	0.48s	0.16s	1.77s	0.32s
n=12	0.60s	128.47s	3.80s	0.38s	37.94s	0.72s
n=14	2.35s	180.06s	15.60s	1.27s	134.57s	4.42s
n=16	9.82s	TO	140.20s	4.18s	1437.01s	10.69s
n=18	41.43s	TO	TO	19.98s	TO	31.09s

表 4. 既存ツールと提案手法の比較（shift）

	Acacia+			提案手法		
	n分割	2 分割	3 分割	n分割	2 分割	3 分割
n=10	48.85s	42.76s	43.22s	0.13s	0.09s	0.09s
n=12	TO	TO	530.58s	0.40s	0.07s	0.11s
n=14	TO	TO	TO	1.03s	0.55s	0.29s
n=16	TO	TO	TO	4.58s	2.52s	2.26s
n=18	TO	TO	TO	24.02s	11.37s	11.24s
n=20	TO	TO	TO	123.54s	61.60s	60.30s
n=22	TO	TO	TO	520.35s	274.33s	278.37s

5. おわりに

本研究では，形式仕様の実現可能性判定を効率的に行うための分割検証法を提案した．一括検証との実行時間を比較した結果，提案手法の大幅な効率向上が確認できた．また，既存の分割検証ツールとの実行時間の比較においても提案手法が上回ることを確認した．

今後の課題は，オートマトンのさらなる縮小化について検討することである．また，仕様の分割方法によっても効率向上が見込めるため，効果的な仕様の分割方法についても検討する必要がある．

謝辞

本研究は JSPS 科研費 JP 22K11980 の助成を受けたものです．

参考文献

[1] E. Filiot, et al., Compositional Algorithms for LTL Synthesis, Automated Technology for Verification and Analysis , ATVA2010, pp. 112-127, 2010.

[2] S. Ito, T. Ichinose, M. Shimakawa, et al. Qualitative analysis of gene regulatory networks by temporal logic. Theoretical Computer Science, vol.594, pp.151-179, 2015.

[3] A. Bohy, et al., Acacia+, a Tool for LTL Synthesis, Computer Aided Verification, CAV2012, pp. 652-657, 2012.

[4] SYNTCOMP/benchmarks https://github.com/SYNTCOMP/benchmarks