「マルチメディア，分散，協調とモバイル
(DICOMO2019) シンポジウム」 令和元年7月

実行時requirement enhancementのための ゲーム空間の差分分析アルゴリズム

李 家隆[1] 相澤 和也[1] 鄭 顕志[1,2] 鷲崎 弘宜[1,2] 本位田 真一[1,2]

A Different Analysis Algorithm for Requirement Enhancement at Runtime

Jialong Li[1] Kazuya Aizawa[1] Kenji Tei[1,2] Hironori Washizaki[1,2] Shinichi Honiden[1,2]

1. はじめに

自己適応システムは，外部環境の変化に対し，システム自身が振る舞いを変更するシステムである．制御器合成技術 [1] を利用した自己適応システムにおいて，システムは (1) 変化した環境を検知し，(2) 開発者が想定した要求から開発者の意図に沿って最大限に要求を保証する要求の集合を分析し，(3) 分析された要求集合を保証するように振る舞いを変更する．相澤ら [2] は，ゲーム空間と勝利領域を構築し，保証できる最大限な要求集合を分析する手法と環境変化時の要求緩和分析手法を提案した．しかし，環境から望ましくない動作（開発者が想定してない外部環境から観測された動作）が発生しなくなる場合に，システムはより高いレベルの要求を保証可能 (以下，requirement enhancement という) かどうかを分析するには数分間以上かかる．本研究は，保証可能な要求を効率的に分析するためのアルゴリズムを提案する．評価実験の生産工場モデルにおいて，実行時間の約 99.95 %が削減された．

2. 研究背景

離散制御器合成技術 [1] とは，ソフトウェアシステムが制御可能な動作と観測可能な動作が相互作用するモデル化した環境と安全性などのシステムがみたすべき要求の集合から，環境モデル上要求を満たす仕様モデルを合成する技術である．ここでは，環境モデルと仕様モデルを，離散的な状態遷移マシンの Labelled Transition System(LTS)[3] で表現する．安全性などの要求を時相論理式の一種である Fluent Linear Temporal Logic(FLTL)[4] で記述する．

相澤ら [2] は制御器と外部環境との相互作用を二人対戦型のゲーム [5] として捉え，保証可能な安全性を特定する手法を提案した．更に、環境変化時にゲーム空間 (ゲームを分析する空間) を差分更新することで効率的な実行時分析を実現した．しかし，[1] の手法は，環境変化によって，環境からの望ましくない動作が発生するような場合にしか用いることができず，環境変化によって望ましくない動作が発生しなくなる場合については考えられてこなかった．例えば，荷物運搬システムにおいて，電源提供が不安定によるロボットによる荷物の持ち上げ動作が失敗しうる動作は，電源の安定供給により消えると，ロボットは荷物を落とす事なく，より効率的に動くことが可能になる．このように，環境モデルにおける望ましくない遷移が限定されることで，requirement enhancement する可能性がある．しかしながら，[1] のアルゴリズムの単純な適用ではゲーム全体の再分析が必要となり，requirement enhancement を効率よく実現できない．requirement enhancement の分析はシステム実行中に行われるため，可能な限り計算時間の短縮が求められる．

3. ゲーム空間の差分更新アルゴリズム

本論文では，望ましくない動作が限定される環境変化を対象とする．つまり，環境モデル上に，システムが観測可能な動作を表す遷移が消える変化を扱う．望ましくない動作の限定により，制御器と外部環境との相互作用を表すゲームとの制御器が要求を保証できない状態の集合であるゲーム空間内の勝利領域への影響は局所的であるが，その範囲を正確に把握することは困難である．本アルゴリズムは，ゲーム空間内の影響を受ける可能性のある状態について要求の保証可否を再分析する．影響を受けない状態については分析が行われないため，計算時間を削減することが期待できる．図 1 で示した例を用いて，本アルゴリズムで

1 早稲田大学理工学術院
2 国立情報学研究所

扱う勝利領域について説明する．

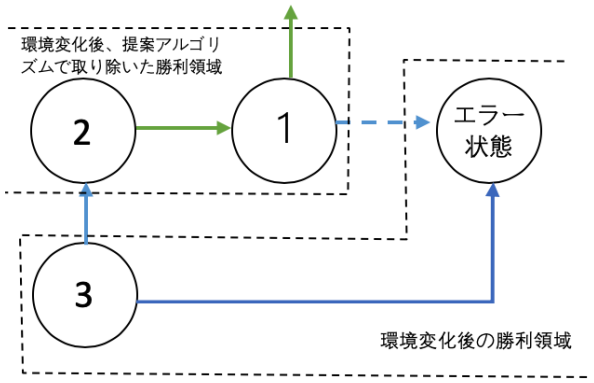

図 1　ゲーム空間内勝利領域の変化の例

図 1 のゲーム空間について説明する．システムはエラー状態に到達すると，この状態に対応する要求を違反する．また，緑の矢印はシステムが制御可能な動作を表す遷移，青い矢印は制御不能で観測可能な動作を表す遷移を意味する．

既存のアルゴリズム [1] で生成される環境側の勝利領域について説明する．環境側の勝利領域は，制御器が制御可能な遷移を選択するだけではエラー状態を回避できない領域である．たとえば状態 1 と状態 3 からは制御不能な遷移によってエラー状態に到達し，状態 2 は制御可能な遷移を選んでもエラー状態に到達する状態 1 に到達してしまう．このような状態は勝利領域に属する．勝利領域の差分特定アルゴリズムは以下の 2 ステップで構成される．

(1) 望ましくない動作の限定よって影響を受ける状態を勝利領域から一旦取り除いた勝利領域 W' を特定する

図 1 の中で，点線の観測可能な遷移が消える場合を例として説明する．本来エラー状態に到達する点線の遷移が消えると，状態 1 から要求を違反するエラー状態に到達できなくなる．よって，状態 1 を勝利領域から取り除く．状態 1 が勝利領域から取り除かれることによって，状態 2 もエラー状態への到達性が不明瞭になり，勝利領域から状態 2 を取り除く．同様に，状態 3 も W から取り除かれる．この例では，勝利領域 W' 内の要素はエラー状態のみとなる．

(2) 勝利領域'W から再度拡張する

(1) では，消えた遷移の影響のみ注目した．勝利領域から取り除いた状態は，他の遷移によって再度勝利領域に属する可能性がある．よって，勝利領域 W' から再度勝利領域拡張する必要がある．拡張の方法は勝利領域の生成方法と同じであるため，勝利領域 W' を初期集合として既存アルゴリズムを利用すれば良い．この例では，勝利領域' 内のエラー状態から再度拡張し，状態 3 が再び勝利領域に入る．

以上の 2 ステップにより，点線で囲まれた部分の差分が勝利領域から取り除かれ，環境変化後の勝利領域が正しく更新される．

4. 評価

評価では，提案アルゴリズムによる計算時間の削減効果とその時間削減の現実的な意味を評価する．実験では，3 つのセルからなるセル生産方式の工場のモデル [6] を利用する．モデル上において，5 つの違う環境変化に対し，既存アルゴリズムと提案アルゴリズムの計算時間を計測した．

既存アルゴリズムの計算時間は平均 140,411，最悪 149,249ms に対し，提案アルゴリズムは平均 69，最悪 86ms である．提案アルゴリズムは，モデルの全状態の平均 0.08%，最悪 0.19%の状態のみ再特定することで，計算時間を平均 99.951%，最悪 99.936%を短縮できた．提案アルゴリズムの差分更新は，計算時間を削減することに有効であることを確認できた．次に，時間削減の現実的な意味を評価する．自己適応システムは，外部環境の変化に迅速に適応することが望まれる．本例題においては，従来アルゴリズムでは計算時間の約 2 分間に対し，提案アルゴリズムでは約 0.05 秒で分析が完了できた．環境変化に対して，迅速に保証可能な要求を特定するために，有意義な時間削減だと言える．

5. おわりに

本論文では，数分間以上かかる requirement enhancement の分析に対して，ゲーム空間の差分に着目した効率的な分析アルゴリズムを提案した．生産工場モデルのスタディーケースにおいて，実行時間の約 99.95 %を削減した事を確認できた．将来の課題として，本手法で再特定した勝利領域を利用し，制御器合成に必要な計算時間も削減することがあげられる．

参考文献

[1] N. D' Ippolito, V. Braberman, N. Piterman, and S. Uchitel, "Synthesis of live behaviour models", FSE ' 10, pp. 77-86, 2010.

[2] K. Aizawa, K. Tei, and S. Honiden, "Identifying safety properties guaranteed in changed environment at runtime", ICA 2018, pp. 75-80, 2018.

[3] J. Magee and J. Kramer, Concurrency: State Models and Java Programs. Wiley Publishing, 2nd ed., 2006.

[4] D. Giannakopoulou and J. Magee, "Fluent model checking for event-based systems," SIGSOFT Softw. Eng. Notes, vol. 28, pp. 257–266, Sept. 2003.

[5] E. Gradel, W. Thomas, and W. Thomas. Automata Logics, and Infinite Games: A Guide to Current Research. New York, NY, USA: Springer-Verlag New York, Inc., 2002.

[6] C. Lewerentz, T. Lindner. Formal Development of Reactive Systems: Case Study Production Cell. Berlin Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, Inc., 1991.