様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　６　月　１９　日現在

機関番号：　１２４０１
研究種目：　基盤研究(C)（一般）
研究期間：　2013～2016
課題番号：　２５３３０００８

研究課題名（和文）時間論理によるリアクティブシステム仕様のプログラム化可能性の判定とプログラム合成

研究課題名（英文）Realizability Decision and Program Synthesis for Reactive System Specification described by Temporal Logic

研究代表者

吉浦　紀晃（YOSHIURA, Noriaki）

埼玉大学・情報メディア基盤センター・准教授

研究者番号：００３０２９６９

交付決定額（研究期間全体）：（直接経費）　　3,700,000円

研究成果の概要（和文）：本研究の成果は、時間論理で記述されたリアクティブシステムの仕様の実現可能性を、推論規則とオートマトンを利用した判定システムにより高速に判定することが可能となり、特に実現可能ではない場合の判定が高速化された。さらに、実現可能ではない仕様を、実現可能に修正するための修正方法として、要求制約式の新たな導出方法を提案した。これにより実現可能な仕様への修正を支援することが可能となった。さらに、仕様から有益な情報を推論するシステムの高速化を行った。
さらに、ネットワークをソフトウェアにより構築する方法であるOpenFlowのネットワークをネットワークトポロジーから構成するための基盤となる検証方法を構築した。

研究成果の概要（英文）：The research proposed the decision method that determinizes the realizabilities of reactive system specifications that are described in temporal logic. The method consists of the inference rules and the automata and can determine the unrealizibilties of the specifications fast. The research proposed the method of deducing constraint formulas of requirement for unrealizable reactive system specifications. The contraint formulas are useful to revise unrealizable reactive system specifications so that they may be realizable. The research also improves the inference system of relevant logic. The inference rule can deduce information from reactive system specifications. The information is useful so that the descriptors of the specifications can verify that the specifications are equal to what the specifications should be. Moreover, the research proposed the verification method of OpenFlow software. The method can be a base of synthesizing OpenFlow software from network topologies.

研究分野：　情報工学

キーワード：　時間論理　モデルチェッキング　実現可能性　リアクティブシステム　適切さの論理

１．研究開始当初の背景

ソフトウェアが社会インフラとなっている今日では、不具合のないソフトウェアの構築やソフトウェアの不具合の検出は極めて重要である。ソフトウェアの不具合にはバッファオーバーフローなどのプログラミングが原因となるもののほかに、ソフトウェアの設計が原因になる場合もある。例えば、DNS(Domain Name Service)におけるキャッシュポイズニングなどはプログラミングの問題ではなくプロトコルやソフトウェアの設計の問題といえる。ソフトウェアの設計は様々なレベルがあるが、ソフトウェア仕様がソフトウェアの基本設計書となる。
一方、社会インフラとなっているソフトウェアのほとんどはリアクティブシステムの制御プログラムである。リアクティブシステムとは飛行機の制御プログラムなど、ユーザとインタラクションをしながらサービスを提供し動作するシステムである。
リアクティブシステムの制御プログラムの構築やその検証のためには仕様が重要であり、仕様を形式的言語で記述することでプログラムの合成や検証を形式的手法により行うことができる。その形式的言語の１つとして時間論理があり、SMV や SPIN など時間論理に基づいたプログラムの検証ツールが提案されている。
現在、SMV や SPIN でプログラムの検証を行うための仕様は簡単なものが対象であるが、今後複雑な記述で表現される仕様を検証する場合、仕様自体に誤りがないかを判定することが重要となる。特に、仕様を満たすプログラムが存在しない場合、プログラムが仕様を満たすか否かを検証すること自体が無駄になる。一方、時間論理により記述されたリアクティブシステム仕様を満たすプログラムが存在するか否か(プログラム化可能性)の判定やプログラムの合成の研究が行われており、Lily などのツールが提案されている。
数値計算プログラムと異なり、リアクティブシステムの制御プログラムの場合、ユーザとのインタラクションを伴って動作し、ユーザの動作を制御することができないため、仕様を安易に記述すると、仕様を満たすプログラムが存在しないこともあり得る。例えば、リアクティブシステムの１つであるエレベータの仕様が以下の要件を持つ場合、仕様を満たすプログラムは存在しない。

- ５階で利用者が「下がるボタン」を押したならば、いつかエレベータが５階に到着する。
- 利用者がエレベータ内の「ドアを開くボタン」を押したならば、ドアは開く。
- ドアが開いている間、エレベータは動かない。

どのようなプログラムでもこの仕様を満たすためには、「ドアを開くボタン」を利用者が押し続ける場合、エレベータは動くことができず、５階にいる利用者が「下がるボタン」を押しても５階にエレベータが到着することはなく、結局仕様を満たすことができない。この仕様には「ドアを開くボタン」が押され続けることがないという暗黙の前提があり、仕様に問題があるとみなせる。このように仕様がプログラム化可能であるか否かを判定することは仕様の不備や暗黙の前提を見つけることでもあり、仕様からプログラムを合成するためにも必要である。
仕様のプログラム化可能性を判定する方法やプログラムを合成する方法は既に提案されている。しかし、これらの方法はオートマトン理論に基づいており、膨大な計算時間とメモリを必要とし、現状では実用的とは言えない。本研究では、論理の特徴である推論規則を考案することで、プログラムの存在判定の高速化を行い実用的なものに近付ける。

２．研究の目的

前述の背景に基づき、本研究は時間論理で記述されたリアクティブシステム仕様を満たすプログラムが存在するか否か、つまりリアクティブシステム仕様(以後、仕様)のプログラム化可能性を高速で判定するための方法を構築し、仕様からのプログラムの合成の方法を構築する。さらにこれらの実装を行う。研究期間内には以下のことを明らかにする。

(1) 時間論理で記述された仕様のプログラム化可能性に関する性質を論理式に対する推論規則を適用することで判定できる証明システムを構築する。仕様 Ψ が時間論理により記述されるとき、Ψ はユーザの動作を示す原子命題とシステムの動作を示す原子命題から構成される。仕様 Ψ がプログラム化可能であることの形式的定義すると次のとおりである。

時間 i のユーザの動作を a_i、システムの動作を b_i としたとき、ユーザの動作列を $a=a_0a_1a_2\ldots$、システムの動作列を $b=b_0b_1b_2\ldots$ とする。また、時間 i のユーザとシステムの動作を合わせたものを (a_i,b_i) とし、その動作列を $(a_0,b_0)(a_1,b_1)\ldots$ とする。仕様 Ψ がプログラム化可能であるとは、すべてのユーザの動作列 a に対して、$(a_0,b_0)(a_1,b_1)\ldots$ が仕様 Ψ を満たすようなプログラム $program$ が存在することである。ただし、$b_i = program(a_0,\ldots,a_i)$ となる。

(2) 前述のプログラム化可能性に関する証明システムが完全性を満たすことは難しいと予想される。つまり、仕様がプログラム化可能または可能でないならば、証明システムが、仕様がプログラム化可能または可能でないと必ず判定することは難しいと予想される。また、完全性を満たすような証明システムが存在しない可能性もある。

証明システムをより完全性を満たすものに近付けるとともに、完全な証明システムの有無について調べる。

(3) プログラム化可能性を満たす仕様や満たさない仕様の特徴を明らかにする。この特徴の有無を調べることで、プログラム化可能性の判定に利用することで高速な判定を可能にする。

(4) 仕様がプログラム化可能ではない場合、その原因の導出方法を構築する。この方法は仕様に問題がある場合、その修正を行うために必要である。

(5) 仕様のプログラム化可能性判定方法とプログラムの合成方法を実装する。

本研究の独創的な点は以下のとおりである。

(1) 従来の研究ではリアクティブシステム仕様のプログラム化可能性の判定やプログラムの合成にオートマトン理論を利用しており、ツールの実装も行われている。しかし、現状では、簡単な仕様しか取り扱うことができない。本研究では仕様のプログラム化可能性に関する証明システムを構築する。この点が従来の研究との違いである。従来の研究では仕様の記述に時間論理を利用しているにも関わらず、プログラム化可能性を判定するために論理本来の特徴の 1 つである証明システムを用いていない。

(2) 仕様のプログラム化可能性に関する証明システムは、仕様がプログラム化可能でない場合に非常に高速に判定できることが明らかになっており、研究代表者は、既存のいくつかの判定器で判定できない仕様のプログラム化可能性を判定している。このように証明システムの構築は判定の高速化につながる。また、証明システムの並行利用や証明戦略の工夫によりさらなる高速化が可能となり、仕様がプログラム化可能ではない場合の原因を導出する方法など新たなツールの開発も可能となる。

(3) 論理の視点から見た場合、本来、証明システムは論理式が正しいことを判定することを目的としているが、本研究の証明システムはプログラム化可能性に関する性質を証明することを目的としており、従来の論理の証明システムとは考え方が異なる。また、完全な証明システムが構築可能であるか否かも時間論理の研究として非常に興味深い。

(4) 本研究により仕様のプログラム化可能性の判定方法の高速化が行われれば、現在利用されつつあるモデル検査手法がより複雑な性質を検証する場合においても、性質自体に誤りがないかどうかを事前に検証することが可能となり、より複雑な仕様の検証を行うことが可能となる。さらに、利用からのプログラムの合成が可能になることで、ソフトウェアの生産性と安全性の向上が期待できる。

３．研究の方法

時間論理で記述されたリアクティブシステム仕様のプログラム化可能性の判定の高速化とプログラムの合成を実現するために、本研究の実施概要は以下のとおりである。

(1) 時間論理で記述された仕様のプログラム化可能性に関する証明システムを構築する。
(2) プログラム化可能な仕様とプログラム化可能ではない仕様の特徴を明らかにする。
(3) 仕様がプログラム化可能ではない場合、その原因を導出する方法を構築する。
(4) 証明システムとオートマトンを利用したプログラム化可能性の判定方法を実装する。
(5) 仕様からプログラムを合成する方法を実装する。

具体的な計画は以下のとおりである。

(1) 時間論理で記述された仕様のプログラム化可能性に関する証明システムの構築

① 仕様がプログラム化可能でないことを判定するための証明システムの構築

時間論理で記述されたリアクティブシステム仕様から推論規則を適用して矛盾が導出される場合、仕様がプログラム化可能ではないと言える証明システムを構築する。これはすでに発表済みであり、証明システムの健全性は示せてはいるが、完全性が明らかではない。また、不足している推論規則があることも予想される。様々な仕様に対して証明システムを適用し、その証明力を検証し、完全性を満たすように証明力を向上させる。

② 仕様がプログラム化可能であることを判定するための証明システムの構築

プログラム化可能であることを示すための証明システムを構築する。具体的には、結論が必ずプログラム化可能となるような証明システムの構築を行う。この証明システムを利用することで、仕様がプログラム化可能であることを高速で判定する方法を構築する。この証明システムは(1)①の場合とは違い、構築できるか否かが明らかではない。初めに簡単な証明システムを構築し、徐々に証明力を持つように証明システムの構築を行う。

(2) プログラム化可能な仕様とプログラム化可能ではない仕様の特徴の分析

① プログラム化可能ではない仕様の特徴の分析

仕様がプログラム化可能ではない場合、仕様から必ず演繹的に推論される式の特徴を分析する。研究代表者は既にいくつかの式の特徴を発見している。このような特徴を見つけるために様々なプログラム化可能ではない仕様を分析する。特徴が得られると、仕様とその特徴を持つ式の否定が矛盾することとなり、恒真性を判定する証明システムを利用することで仕様がプログラム化可能ではないことを判定することが可能となる。

② プログラム化可能である仕様の特徴の分析

プログラム化可能である仕様から必ず演繹的に推論される式の特徴を分析する。(2)①と同様に、様々なプログラム化可能である仕様を分析することで特徴を見つけ出す。得られた特徴から、仕様がプログラム化可能であるか否かを、恒真性を判定する証明システムを利用することで判定することができる。

(3) 仕様がプログラム化可能ではない原因を導出する方法の構築

① 原因の導出方法の構築

仕様がプログラム化可能ではないことを判定する証明システムでは、仕様がプログラム化可能ではないことが証明された場合、その証明は仕様がプログラム化可能ではない原因に関する情報を含むことが分かっている。この情報を利用し、仕様がプログラム化可能ではない原因を導出する方法を構築する。

② 最適な原因の選択方法の構築

仕様がプログラム化可能ではない場合、その原因は 1 つとは限らない。(3)①で構築される導出方法が、仕様のプログラム化可能ではない原因を複数個導出することもある。複数の中から原因として最適なものを選択または導出する方法を構築する。最適な原因を決定するために、原因の間に順序関係を導入する。また、(3)①の導出方法がすべての原因を導出できるか否かを分析する。

(4) 証明システムとオートマトンを利用した判定方法の実装

① 証明システムの推論規則適用のための証明戦略の構築

(1)で構築したプログラム化可能性に関する 2 種類の証明システムの実装を行う。証明システムの実装のためには、証明システムの推論規則をどのように適用するかの方針、つまり証明戦略を構築する必要がある。証明を効率よく行うとともに証明システムの能力を最大限引き出せるように証明戦略を構築する。

② 証明システムの実装

(4)①で構築した証明戦略に基づき、証明システムを実装する。実装においては、マルチスレッドによる並列化など実装上の工夫により高速化を図る。

③ 仕様がプログラム化可能ではない原因の導出方法の実装

(4)②におけるプログラム化可能ではないことを判定する証明システムの実装を利用して、仕様がプログラム化可能ではない原因を導出する方法を実装する。最適な原因を選択または導出する方法も実装する。

④ オートマトンに基づく判定方法の実装

証明システムの完全性が保証されない場合、仕様からオートマトンを構成し、オートマトンを調べることでプログラム化可能性を判定する。この方法自体は既に確立されており、証明システムとオートマトンを併用した形での判定システムを提案している。証明システムを補完する形でオートマトンによる判定方法を実装する。

(5) 仕様からのプログラムの合成方法の実装

① 証明図からのプログラムの合成

証明システムにより仕様がプログラム化可能であると判定された場合、その証明図からプログラムを合成する方法を確立し、その実装を行う。

② オートマトンからのプログラムの合成

仕様のプログラム化可能性をオートマトンで判定した場合、プログラムの合成をオートマトンから行う方法を実装する。

③ 最適なプログラムの選択方法の実装

仕様から合成されるプログラムは複数個存在する場合がある。その中から最適なプログラムを選択する方法を構築し実装する。

４．研究成果

本研究では、時間論理で記述されたリアクテ

ィブシステムの仕様から、プログラムが合成可能であるかを判定するための高速化を目指し、以下の研究成果を得た。

(1) 推論規則による強充足可能性や段階的充足可能性の判定手続きの構築

時間論理で記述されたリアクティブシステムの仕様を満たすプログラムが存在するか否か、つまり、仕様の実現可能性を判定することは非常に大きな計算量を必要とする。計算量自体を減少させることにより、判定手続きの高速化を行うことは難しい。そこで、強充足可能性や段階的充足可能性を満たさない場合に特化した判定方法として、推論規則を用いた判定手続きを既に提案していたが、この改良を行った。この推論規則による判定は、従来のオートマトンを利用するものではなく、推論規則の利用により、強充足可能ではないことや段階的充足可能ではないことを示すものであり、高速に判定可能である。しかし、判定できない場合もある。本研究では、従来の推論規則よりも証明力が強い推論規則を構築した。

(2) オートマトンと推論規則の融合による実現可能性の高速化

時間論理で記述されたリアクティブシステムの仕様の実現可能性を判定する方法としては、オートマトンを利用する方法が一般的である。オートマトンを利用すれば、判定が可能であるが、計算量が大きくなる。そこで、オートマトンの利用に加え、推論規則による強充足可能性や段階的充足可能性の判定を同時に行うことで、仕様が実現可能ではないときには、高速に判定が可能となった。

(3) 構文の制限による実現可能性の性質の特徴づけ

時間論理で記述されたリアクティブシステムの仕様の構造、つまり、論理式の形から実現可能性の判定が可能ならば、オートマトンの生成や推論規則の利用と合わせて、より高速に実現可能性に関する判定が可能となる。本研究では、実現可能性と強充足可能性や段階的充足可能性が同一になる場合の仕様の形を明らかにした。そして、仕様の実現可能性を判定するときに、仕様の形から、実現可能性と強充足可能性や段階的充足可能性が同一になるかを判定し、この結果に基づき、強充足可能性や段階的充足可能性の判定によって、実現可能性の判定を行うことが可能となり、高速に実現可能性の判定を行うことができるようになった。

(4) 制約式の導出方法の提案

時間論理で記述されたリアクティブシステムの仕様が実現可能ではない場合、仕様を実現可能となるように仕様を修正することが必要になる。そのための方法の一つとして、要求制約式の新しい導出方法を提案した。リアクティブシステムの仕様が実現可能ではない場合、利用者の利用方法によって、リアクティブシステムが利用者の要求に応えることができないような仕様となっている。利用者が、この利用方法を行わなければ、リアクティブシステムは要求に応じることができ、すなわち実現可能であるとみなすことができる。このリアクティブシステムが要求に応えることができない利用者の利用方法を表す論理式が要求制約式である。この要求制約式は、リアクティブシステムの仕様が実現可能ではない理由となり、仕様を実現可能に修正するためには重要な情報である。この要求制約式の導出はその精度と計算量の間にトレードオフがあるが、本研究では、より精度の高い要求制約式を計算量のオーダーを変えずに導出することを可能とした。

(5) 適切さの論理の証明システムの高速化

時間論理で記述されたリアクティブシステムの仕様に基づき、リアクティブシステムを構築する場合、その仕様が仕様記述者の意図通りであるかを判断する必要がある。この判定は、仕様記述者が仕様を読んで判定するしか方法がない。そこで、仕様から、仕様が含む情報のうち有益な情報を推論し、仕様記述者へ提供することにより、仕様が仕様記述者の意図通りであるかを判定することの支援が可能となる。この有益な情報を推論する方法として、適切さの論理を用いる方法がある。本研究では、この推論システムの高速化を行った。

(6) ネットワークプロトコルや OpenFlow プログラムの検証

ネットワークをソフトウェアにより構築する Software Defined Network (SDN)の一つに Open Flow がある。OpenFlow は実際に Google でのネットワーク構築で利用されている。ネットワークを構築するソフトウェアの検証方法にはモデルチェッキングなどの方法があるが、Coq を利用した検証方法も提案されている。Coq を利用した検証方法では、ネットワーク機器が単一のみの場合の検証が可能であるが、本研究では、複数のネットワーク機器でも検証可能となる検証方法を提案した。この検証方法を利用することで、ネットワークトポロジーを与えるだけで、基本的な OpenFlow プログラムを構成することができる。また、ネットワークプロトコルの安全性の検証を行う方法を提案した。

(7) ソフトウェアのデータレースの検出

ソフトウェアにおけるデータレースを検出する方法は提案されているが、動的にソフトウェアが構築される場合、つまり、インターネット越しに、ソフトウェアに必要とされるライブラリを取得する場合のデータレース検出は行われていなかった。本研究では、動的に必要とされるライブラリの候補を、ソフトウェアと合わせて分析することで、起きる可能性のあるデータレースを検出することが可能となった。

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計１件）

① 魏偉, 吉浦紀晃, 文字列解析の利用による動的クラスローディングに対応した静的データレース検出方法, 情報処理学会論文誌, Vol.54, No.2, pp.787-796 (2013) (査読有)

〔学会発表〕（計 ８ 件）

① Noriaki Yoshiura, The Relation Between Syntax Restriction of Temporal Logic and Properties of Reactive System Specification. 9th Asian Conference on Intelligent Information and Database Systems, Lecture Notes in Computer Science 10192, pp.105-114, Kanazawa Japan (2017) (査読有)

② Hiroaki Date, Noriaki Yoshiura, Computational Verification of Network Programs for Several OpenFlow Switches in Coq,16th International Conference of Computational Science and Its Applications, Lecture Notes in Computer Science 9787, pp.223-238, Beijing China (2016)(査読有)

③ Yuichi Fukaya, Noriaki Yoshiura, Extracting Environmental Constraints in Reactive System Specifications, 15th International Conference on Computational Science and Its Applications, Lecture Notes in Computer Science 9158, pp.671-685, Banff Canada (2015)(査読有)

④ Noriaki Yoshiura, Yuma Hirayanagi, Implementation of Decision Procedure of Stepwise Satisfiability of Reactive System Specifications. 15th International Conference on Computational Science and Its Applications, Lecture Notes in Computer Science 9158, pp.686-698, Banff Canada (2015)(査読有)

⑤ Noriaki Yoshiura, Implementation of Theorem Prover of Relevant Logic, 28th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Lecture Notes in Computer Science 9101, pp.13-22, Seoul Korea (2015)(査読有)

⑥ Noriaki Yoshiura, Wei Wei, Static Data Race Detection for Java Programs with Dynamic Class Loading, 7th International Conference on Internet and Distributed Computing Systems, Lecture Notes in Computer Science 8729, pp.161-173, Amantea, Italia (2014)(査読有)

⑦ Noriaki Yoshiura, Verification of System Requirements, 4th International Mechanical Engineering Research Conference, Cebu City Philippines (2014) (Keynote Speaker)

⑧ Natsuki Kimura and Noriaki Yoshiura, Construction and Verification of Mobile Ad Hoc Network Protocols, In Proceedings of 5th International Symposium on Cyberspace Safety and Security, Lecture Notes in Computer Science, Vol.8300, pp.198-212, Zhangjiajie China (2013)(査読有)

〔図書〕（計 0 件）

〔産業財産権〕

○出願状況（計　0　件）

○取得状況（計　0　件）

〔その他〕
なし

６．研究組織
(1)研究代表者
吉浦　紀晃（YOSHIURA, Noriaki）
埼玉大学・情報メディア基盤センター・准教授
研究者番号：00302969

(2)研究分担者
なし

(3)連携研究者
なし

(4)研究協力者
なし