モデル検査手法を用いた組込みアセンブリプログラムのリアルタイム性の形式的検証

学　　位　　論　　文　　要　　旨

モデル検査手法を用いた組込みアセンブリプログラムのリアルタイム性の形式的検証

Formal Verification of Real-Time Properties for Embedded Assembly Program Using Model Checking

金沢大学大学院自然科学研究科
電子情報科学専攻

呉　亜軍 (WU YAJUN)

ABSTRACT

In recent years, embedded systems have been widely used in autonomous car, national defense, medical equipment and IoT (Internet of Things). It is important to verify real-time properties for embedded systems, not only the logical validity. The real-time property of embedded system has to be tested extensively and validated because errors may lead to severe or even fatal events, as in the brake instructions of the autonomous car to the execution time are very important and must be executed within the deadline. Therefore, to verify the real-time properties is the key problem for the design of embedded real-time systems.

To verify real-time properties of embedded systems, we verify assembly program rather than C program. First of all, interrupt processing can be performed for each one instruction. Second, the interface part with the hardware has the assembly description part. Then, it is possible to compute the execution time accurately. In this thesis, we can verify real-time properties by proposing the timed Kripke structure. We develop the simulator to generate timed Kripke structure from assembly program and implement the simulator of the robot's processor to be verified. In this approach, to verify the real-time property of assembly program, we propose the timed Kripke structure which extends Kripke structure by includes the execution time. The property to be verified is described by RTCTL. We will describe model checking using RTCTL (Real-Time CTL) in order to verify whether the timed Kripke structure satisfies real-time properties. In this thesis, we propose RTCTL model checking algorithm after generating timed Kripke structure. Also, we implement a model checker in order to verify whether the timed Kripke structure satisfies real-time properties.

On the other hand, when performing model checking of large-scale embedded systems, the technique of abstraction is used as a method to reduce the number of states. Guaranteeing real-time performance is important for embedded systems, and reducing the number of states is also important. In this thesis, we also propose verifying real-time properties of assembly program based on lazy abstraction and refinement. In particular, we propose one method for modeling assembly programs with time and another method for refining abstract models using Interpolation. Finally we show the effectiveness of our proposed method by verification experiments.

要旨

組込みシステムは IoT(Internet of Things)，航空・宇宙や国防などで広く使用されている．特に近年，自動運転を始め，機械学習など先進技術をより一層取り入れている．システムのより高い安全性や信頼性が保証されることは必要不可欠である．そして，システムの安全性を保証する手法としてモデル検査がある．モデル検査では，対象のシステムをモデル化し，そのモデルの状態に対して網羅的に探索することで安全性などの検証を行う．網羅的に探索するため，人の手では発見しにくいバグやシステムの欠陥なども見つけ出すことができる．

組込みシステム安全性保証の研究は，社会的かつ科学技術上の重要な国際的課題である．組み込みシステムに対して，論理的正当性だけではなく，リアルタイム性の検証も重要である．例えば，最近注目が集めている車の自動運転では，センサが車の前に障害物を検出し，そこで減速始めるかどうかを重要であるが，どのくらいの時間内に減速を始めるのも重要である．リアルタイム性検証の考慮がなければ，たとえそれが何万回に 1 回起こるかどうかであるとしても，そのシステムは非常に危険であると言える．

論文の 2 章 (要旨の 1 章) では，ハードウェアとの相互作用及びタイミング制約を検証するために，アセンブリプログラムを形式的意味モデルという時間付き Kripke 構造（アセンブリ命令の実行時間付き状態遷移システム）に変換し, リアルタイム性のソフトウェアモデル検査手法を提案する．組込みシステムのリアルタイム性を検証するために、C プログラムではなくアセンブリプログラムを検証する．まず，アセンブリプログラムが命令ごとに割込み処理が発生する．さらに，ハードウェアとのインタフェース部分にはアセンブリ記述部分がある．最後に，精確な実行時間を計算することができる．そして，本研究は先行研究が開発したシミュレータを拡張や改良し，動的プログラム解析を用いることで，アセンブリプログラムを網羅的に探索することにより，各命令の実行時間を計算することができる．そして，時間付き Kripke 構造を生成する．検証対象となるロボットのプロセッサをベースにシミュレータを実装し，RTCTL モデル検査によるリアルタイム性の検証を行った．

組込みシステムの検証では，検証効率の向上と状態爆発の回避の両方が重要である．大規模な組込みシステムのモデル検査を行う際には，状態数を減らす方法とし，抽象化の手法が用いられる．また，リアルタイム性の検証も重要である．そこで，論文の 3 章（要旨の 2 章）はアセンブリプログラムのリアルタイム性を検証するために，抽象化精錬と Interpolation に基づいた手法を提案する．そのために，リアルタイム性を検証可能にアセンブリプログラムをモデル化する方法と，補間を用いて抽象モデルを精錬化する方法を提案する．最後に，提案手法の有効性を検証実験により証明する．

第1章　リアルタイム時相論理RTCTLを用いた組込みアセンブリプログラムのモデル検査

1.1　時間付きKripke構造

モデル検査を行う際に，システムを表現するモデルの状態空間を網羅的に探索するために，Kripke構造を用いることが多い．

本研究では，計算モデルであるKripke構造を時間関数 （TM）により拡張した時間付きKripke構造を定義する．

Definition 1 (Timed Kripke structure) *Timed Kripke structure is* $M = (S, S_0, R, L, TM)$.

- S　状態の有限集合．
- $S_0 \subseteq S$　初期状態の集合．
- $R \subseteq S \times S$　状態集合間の遷移関係．
- $L : S \to 2^{\mathrm{AP}}$ は各状態における成り立つ原子命題の集合を割り当てるラベリング関数．AP は原子命題の集合である．
- $TM : S \to N$ は各命令の実行時間を各状態に割り当てる時間関数．Nは自然数である．

1.2　リアルタイム時相論理RTCTL

本研究はリアルタイム性を検証するため，リアルタイム時相論理RTCTL(Real-Time Computational Tree Logic) を定義する．

Definition 2 (Syntax of RTCTL) *RTCTL* 式の構文は次のとおりである．

- 各原子命題 *AP* は *RTCTL* 論理式．
- p, q が *RTCTL* 論理式であれば，$p \wedge q$ と $\neg p$ は *RTCTL* 論理式．
- p, q が *RTCTL* 論理式であれば，$E(p\ U\ q), A(p\ U\ q), EXp$ は *RTCTL* 論理式．
- p, q が *RTCTL* 論理式かつ k が任意の自然数であれば，$E(p\ U^{\leq k}\ q)$ と $A(p\ U^{\leq k}\ q)$ は *RTCTL* 論理式．

実行時間を表す自然数の時間制約を用いたことにより，リアルタイム性の性質を記述できる．

他のRTCTL論理式は，次のように定義される．

$\mathrm{AF}^{\leq k}\ \mathrm{q} = \mathrm{A}(\mathrm{true}\ \mathrm{U}^{\leq k}\ \mathrm{q})$　　$\mathrm{EF}^{\leq k}\ \mathrm{q} = \mathrm{E}(\mathrm{true}\ \mathrm{U}^{\leq k}\ \mathrm{q})$

$\mathrm{AG}^{\leq k}\ \mathrm{p} = \neg\mathrm{EF}^{\leq k}\neg\mathrm{p}$　　$\mathrm{EG}^{\leq k}\ \mathrm{p} = \neg\mathrm{AF}^{\leq k}\neg\mathrm{p}$

Definition 3 (Semantics of RTCTL) 本研究の $E(p\ U^{\leq k}\ q)$ と $A(p\ U^{\leq k}\ q)$ 論理式は時間付き *Kripke* 構造 $M = (S, S_0, R, L, TM)$ 上に定義する．

(1) $E(p\ U^{\leq k}\ q)$

時間付き *Kripke* 構造 M 上のある状態列 $s_0, s_1, \ldots, s_j, \ldots, s_i, \ldots$ に対し，$\exists i\ 0 \leq i$ を満たすと，

$$s_i \models q \text{ かつ } \sum_{\alpha=0}^{i} TM(s_\alpha) \leq k \text{ かつ } \forall j\ 0 \leq j < i,\ s_j \models p$$

となる．

(2) $A(p\ U^{\leq k}\ q)$

時間付き *Kripke* 構造 M 上すべての状態列 $s_0, s_1, \ldots, s_j, \ldots, s_i, \ldots$ に対し，$\exists i\ 0 \leq i$ を満たすと，

$$s_i \models q \text{ かつ } \sum_{\alpha=0}^{i} TM(s_\alpha) \leq k \text{ かつ } \forall j\ 0 \leq j < i,\ s_j \models p$$

となる．

本研究で向き合う課題は各状態の実行時間を精確に計算するこどによりリアルタイム性を検証することである．そこで本研究は動的プログラム解析を用い，精確な実行時間を含める時間付き Kripke 構造を生成する．

1.3 検証システム

本研究の検証システムは図 1.3.1 で示す．

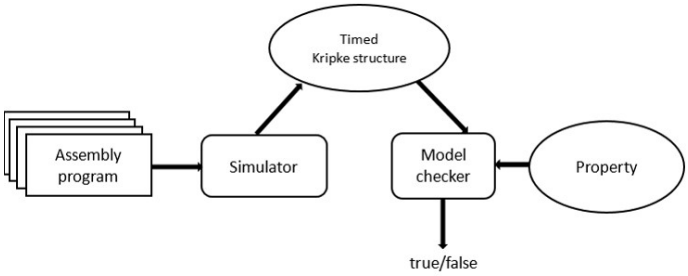

図 1.3.1: 検証システム

一般的に，時間付き Kripke 構造は有限状態であるため，本研究のアプローチは有効である．検証システムについて以下に説明する．

(1) 最初に，実行可能な割り込み処理が存在するかどうか判断しする． 実行可能な割り込み処理が存在する時，割り込み処理を実行する．その後，割込み命令を実行するとともに状態を生成する．

(2) 続いて，実行可能な割り込み処理が存在しない場合は，命令を実行し，状態を生成する．

(3) 最後に，すべての状態が生成された後に RTCTL モデル検査を行う．

以下の例に示すように，状態 t から状態 s への遷移が存在する場合．$E(pU^{leqk}q)_{state\ s}$ が状態 s で true であることを示しているので，不動点定理により次のような式で表すことができる．

$$E(pU^{\leq k}q)_{state\ s} = q_{state\ s} \vee (p \wedge EXE(pU^{\leq k}q))_{state\ t}$$

図 1.3.2 に示すように，検証はバックワードで行われる．状態 s の遷移元状態 t を実行時間の計算により，状態 t にラベルを付けるとリアルタイム性の検証ができる．タルスキー不動点定理を用いることにより，提案手法の有効性を証明できる．

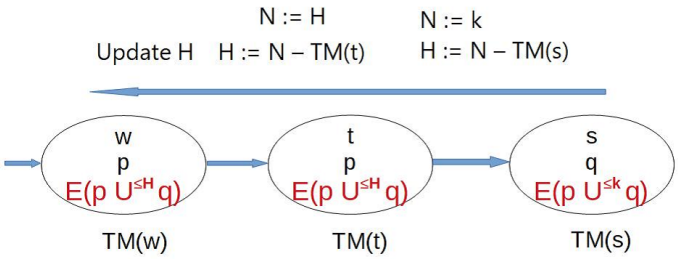

図 1.3.2: verfication transition

1.4 RTCTL 論理式の不動点による定義

次に，RTCTL 論理式を不動点で表現する．

Definition 4 (Fixpoints representation of RTCTL formulas) *RTCTL* の不動点表現は $E(p\ U^{\leq k}\ q)$ と $A(p\ U^{\leq k}\ q)$ 論理式の不動点表現を時間付き *Kripke* 構造 $M = (S, S_0, R, L, TM)$ 上で定義する．

(1) $E(pU^{\leq k}q)$ の不動点表現

$E(pU^{\leq k}q)_{state\ s} = \mu Z.q \vee (p \wedge EXZ)_{state\ t}$ により，

$E(pU^{\leq k}q)_{state\ s} = q_{state\ s} \vee (p \wedge EXE(pU^{\leq k}q))_{state\ t}$ である．

ここで，$p \wedge EXE(p\ U^{\leq k}\ q) = E(p\ U^{\leq r}\ q)$. q を成り立つ状態 s_q に対して，$r = k$ - $TM(s_q)$.

(2) $A(pU^{\leq k}q)$ の不動点表現

$$A(pU^{\leq k}q)_{state\ s} = \mu Z.q \vee (p \wedge AXZ)_{state\ t} \text{ により，}$$
$$A(pU^{\leq k}q)_{state\ s} = q_{state\ s} \vee (p \wedge AXA(pU^{\leq k}q))_{state\ t} \text{ である．}$$

ここで，$p \wedge AXA(p\ U^{\leq k}\ q) = A(p\ U^{\leq r}\ q)$. q を成り立つ状態 s_q に対して，$r = k$ - $TM(s_q)$.

最小不動点が存在することにより，アルゴリズムに沿ってバックワードに検証し，時間制約 k が 0 以下になるとリアルタイム性を満たせないと判断し，検証を終了する．E(p $U^{\leq k}$ q) のモデル検査アルゴリズムは，不動点定理に基づいて証明できる．

1.5 検証実験

本研究はマイコン H8/3687 を基づいてに検証対象プログラムを用い，検証実験を行う．実行時間はアセンブリ命令の実行ステート数から求める．

1.5.1 Program 1

Program 1 (target_motor) は，三つのセンサの値を取得し，その値からロボットの左右にある二つのモータを制御し，ロボットを動作させる．このセンサは白と黒を識別可能である．初期化後，タイマ b 1 のタイマオーバーフロー割込みが発生するとセンサの値を取得し，新しい目標電流値を設定する．タイマ v のタイマーオーバーフロー割込みが発生すると電流目標値と測定した電流値から PID 制御を行い，その値をモーターに出力する．

ロボットは三つのセンサ値により，表 1.5.1 示したようにモータの速度を制御行い，センサの値からロボットの右寄りの状態 (M_1) を少し左へ曲がる (モーター L の速度を 30%から 60%に上げる)，そして少し右寄りの状態 (M_2) を左へ曲がる (モーター L の速度を 60%から 100%に上げる)，中央の状態を直進などに制御できる．

DC (direct current) モータ速度制御のリアルタイム性を検証するため，以下のようなプログラムを検証対象として，ロボットが DC モータの速度制御によって，右寄りの状態 (M_1) から少し右寄りの状態 (M_2) になる，さらに直進 (M_3) になる，続いて少し左寄りの状態 (M_4) になる，次に左寄りの状態 (M_5) になる，最終は直進 (M_3) に戻すというプログラムである．同時に，表 1.5.2 のように六つの RTCTL 論理式を用意し，実験を行う．

RTCTL1 - 6 中の M_1 - M_6 は表 1.5.1 を示したように，モータの回転速度の目標値を示し，レジスタ ER1 の値で示される．RTCTL 1 はあるパスにおいて時間制約 k_1 以内に，モータの回転速度が目標値に満たす．RTCTL 2 はあるパスにおいて時間制約 k_1 以内にモータの回転速度が目標値に満たす，かつあるパスにおいて時間制約 k_2 以内にモータの回転速度が二回目の目標値に満たす．RTCTL 3 - 6 は類同のため，ここで省略する．

今回の実験はタイマ b1 が 1024μs ごとに割り込み発生する，タイマ v は 400μs ごとに割り込みを発生する．モータを速度制御のため，目標電流値はタイマ b1 が割り込み発生する時に設定する，その後，タイマ v が割り込み発生する時に目標値と測定した電流値から PID 制御を行う，その値をモーターに出力する．今回の時間制約を以下値のように設定する．

$$k_1 = k_2 = k_3 = k_4 = k_5 = k_6 = 30000\ (1500\mu s)$$

検証結果は表 1.5.3 で示す．false になる例は RTCTL 3 を対象として，Program 2 の中に二回目タイマ b1 割り込み発生する時，delay コードを追加して作成したものである．本実験により，検証性質の RTCTL 2 - 7 は，モータの回転数制御の回数が増加するにつれ，状態数が増加する．マイコンの初期化を行っているため，状態数よりも遷移数が少ない．最後に，RTCTL 3 を対象として false になる例から，時間制約内にモータの回転速度が目標値に満たすかどうかを検証できた．今回の実験は DC モータ速度制御のリアルタイム性の検証の有効性を示した．

表 1.5.1: The situation of speed control

motor(state)	sensor	motorL	motorR	robot ' s situation
M_1	100	30%	90%	右寄り
M_2	110	60%	90%	少し右寄り
M_3	010	100%	100%	中央
M_4	001	90%	30%	少し左寄り
M_5	011	90%	60%	左寄り
M_6	010	100%	100%	中央

表 1.5.2: RTCTL formulas of Program 1

RTCTL	formula
RTCTL 1	$\mathrm{EF}^{\leq k_1}(M_1 = speed) = \mathrm{E}(\mathrm{true}\ \mathrm{U}^{\leq k_1}(M_1 = speed))$
RTCTL 2	RTCTL 1 $\wedge$ $\mathrm{E}(\mathrm{true}\ \mathrm{U}^{\leq k_2}(M_2 = speed))$
RTCTL 3	RTCTL 2 $\wedge$ $\mathrm{E}(\mathrm{true}\ \mathrm{U}^{\leq k_3}(M_3 = speed))$
RTCTL 4	RTCTL 3 $\wedge$ $\mathrm{E}(\mathrm{true}\ \mathrm{U}^{\leq k_4}(M_4 = speed))$
RTCTL 5	RTCTL 4 $\wedge$ $\mathrm{E}(\mathrm{true}\ \mathrm{U}^{\leq k_5}(M_5 = speed))$
RTCTL 6	RTCTL 5 $\wedge$ $\mathrm{E}(\mathrm{true}\ \mathrm{U}^{\leq k_6}(M_6 = speed))$

表 1.5.3: The results of verifying Program 2

RTCTL	states	relations	verification time(s)	execution time(ms)	result (ms)
RTCTL 2	7249	7077	238.2	1.3062	true
RTCTL 3	13402	13172	625.7	2.6186	true
RTCTL 4	17570	17283	866.7	3.5312	true
RTCTL 5	23725	23380	1245.3	4.8438	true
RTCTL 6	29878	29475	1838.0	6.1565	true
RTCTL 7	34047	33587	2274.2	7.0687	true
RTCTL 3	16146	15852	754.0	3.0551	false

1.6 おわりに

本研究では，リアルタイム性モデル検査の検証システムを提案した．その中，RTCTL モデル検査器は，時間付き Kripke 構造がリアルタイム性を満たすかどうかを検証できた．

今後の課題として，CEGAR(counterexample-guided abstraction refinement) を用いた組込みシステムのリアルタイム性モデル検査を研究する．この他にも，組み込みシステム他の性質の検証と考えている．リアルタイム性モデル検査の検証結果が false になる時，その理由を同時に出力できるようにする．リアルタイム性が満たさないあるいは安全性や到達可能性を満たさないかはっきり分かるようにする．今回の検証システムはシミュレータが状態生成後に RTCTL モデル検査行うですが，シミュレータが状態生成しながら RTCTL モデル検査の研究がある．シミュレータが状態生成しながらのモデル検査では無限状態を検証可能にや膨大なシステムの有効性を研究続ける．

第2章　抽象化精錬とInterpolationを用いた組込みアセンブリプログラムにおけるリアルタイム性モデル検査

2.1　抽象化

本研究では Lazy abstraction という抽象化手法を用い，抽象化を行う．

入力のアセンブリプログラムに対して，図 2.2.1 のようにプログラム解析を行う．まず入力のプログラムに BYACC と JFlex というツールを用いて構文解析と字句解析を行い，解析されたコートとメモリマップから抽象遷移関係を求めるための CFA(Control Flow Automata) を生成する．そして抽象化を行う．

時間付き Kripke 構造の状態集合は有限であっても非常に大きな集合となり，モデル検査が非常に困難の場合はある．そのため，大規模なプログラムに対してモデル検査を行う場合，抽象化により状態数を減らすことが必要となってくる．そこで，本研究では，図 2.1.1 の抽象化精錬 (CEGAR : CounterExample Guided Abstraction Refinement) を用いている．

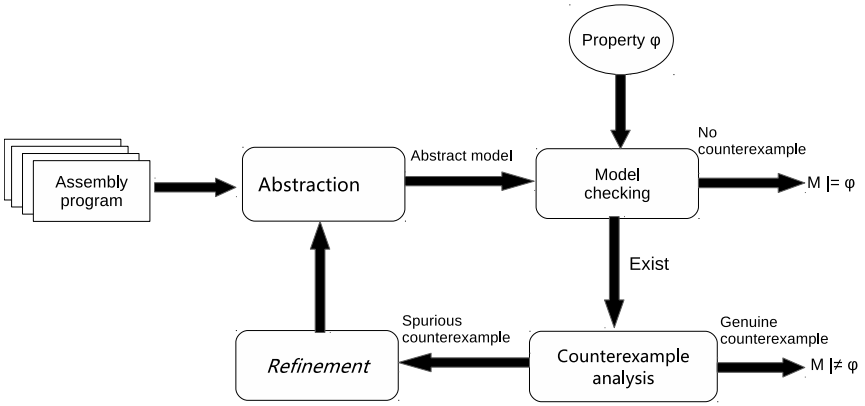

図 2.1.1: CEGAR

2.2　時間付き抽象到達木

すべてのパスが初期状態から時間制約内に到達することを検証するために，本研究は先行研究が提案した Timed ART を 2 章の時間付き Kripke 構造 M = (S, S_0, R, L, TM) 上で構築する．

ART はプログラムの到達可能状態空間の一部を表す木である．一つの ART パスは一つのプログラムの実行パスを表す．

Definition 5 (時間付き抽象到達木) *Timed ART (Timed Abstract Reachability Tree)* は初期状態から全てのパスで，時間制約以内にある状態に到達することを表す木である．*Timed ART* は以下となる．

- *Timed ART* のパスはプログラムの初期状態から始まる．

- *Timed ART* 中のすべての状態 *s(loc, post, pre, pred, tm)* において，到達可能領域 *pred* かつ実行時間 *tm* は満たすことができる．つまり, 状態は有限であり，事後条件によって到達木がこれ以上に伸びないである．

Timed ART の状態 *s : (loc, post, pre, pred, tm)* が時間付き *Kripke* 構造 M における状態集合 S 内の各状態のプログラムロケーション *(program-counter* 値*)* を *loc* とする．遷移関係集合 R からもし遷移条件プログラムオペレーション *op* により $loc \xrightarrow{op} loc'$ であれば，到達可能領域 *pred* に到達可能とすると遷移先エッジ *post (pred, op)* となる．そして，エッジ *(loc, loc')* に対して，*post (pred, op)* $\Rightarrow$ *pred'* となる．同様に遷移条件プログラムオペレーション op_0 により $loc_0 \xrightarrow{op_0} loc$ であれば，遷移元エッジ *pre* $(op_0, pred_0)$ である．さらに，時間付き *Kripke* 構造 M の TM 関数から初期状態から現状態までの総実行時間 *tm* を計算できる．そして，エッジ *(loc, loc')* に対して，実行時間 *tm* を満たすとなると $loc \xrightarrow[tm']{op} loc'$ となり，遷移先エッジ *post (pred, op, tm')* となる．

s:Timed ART のノード
loc: CFA におけるプログラムロケーション
post: 遷移先エッジ
pre: 遷移元エッジ
pred: データ状態の集合を表す *bool* 式 *(*到達可能領域*)*
tm: 初期状態から現状態までの総実行時間

2.3 Lazy Abstraction

本研究では，検証モデルの一部分に必要な述語のみを追加することで状態数の削減を行う Lazy Abstraction を用い，抽象化された時間付き Kripke 構造を生成する．Lazy Abstraction では，述語を発見するたびに抽象モデル全体を再構築する必要がないため，抽象モデルを構築する時間は Predicate Abstraction を用いた場合に比較すると高速になる．Predicate Abstraction では，新しい述語を発見するたびに抽象モデル全体をその述語で抽象化する．そのため述語の数が増えるにつれ，状態数も増加していくため，巨大なプログラムでは状態爆発が発生する可能性がある．

Lazy Abstraction は前方エラー探索 (Foward Search) と後方反例解析 (Backwards Counterexample Analysis) による 2 つのフェーズで構成される．前方エラー探索フェーズでは，図 2.1.1 の抽象化およびモデル検査の処理を行い，後方反例解析フェーズでは反例解析および述語精錬を行う．

本研究では，新しい提案部分は抽象化により生成された抽象時間付き Kripke 構造から，前方エラー探索の Timed ART（時間付き抽象到達木）の構築と後方反例解析に Craig's Interpolation を用い，精錬を行い部分である．そして，SMT ソルバを用いることにより Interpolants を求める．そこで，SMT ソルバ Z3 の機能の一部である iZ3 を用いて Interpolation による反例解析を行う．

2.4 検証システム

本研究では，アセンブリプログラムのリアルタイム性を検証する．高速化検証のため，抽象化と精錬を用いた．そして，アセンブリプログラムをモデル化し，Interpolation を用いて抽象モデルを精錬するリアルタイム性を検証する検証システムを提案する．

図 2.4.1 のように，検証システムを示す．

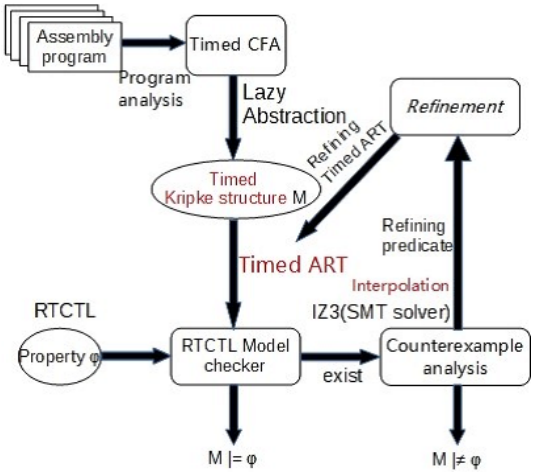

図 2.4.1: Verification system

まず入力のアセンブリプログラムに対して，図2.1.1のようにプログラム解析を行ウ．プログラム解析では，Parser/Lexer（Java用の字句解析器生成器JFlexとパーサ生成器BYACC）を用いて，アセンブラプログラムの字句解析や構文解析を行う．そして，各アセンブリ命令を解析し，アセンブラ制御命令によるアドレス空間の確保・指定に基づいてメモリマップを作成する．その後，命令の実行時間を考慮したTimed CFAを生成する．生成されたTimed CFAに対して，Lazy abstractionという抽象化手法を用いることにより抽象時間付きKripke構造を生成する．抽象時間付きKripke構造を基についてTimed ARTを構築する．続いて，Timed　ARTに対して，RTCTLを用いて，記述された検証性質を満たすかどうかをモデル検査を行う．反例がないと検証性質を満たす．反例がある場合はiZ3というSMTソルバを用いて反例解析を行い．実反例だど、性質を満たさない．偽反例の場合，interpolationを用いたiZ3を精錬述語を生成する．そして，Timed ARTに精錬を行う．再びモデル検査を行い，以上の手順を繰り返す．

本研究では，Craig's interpolationを用いて，反例解析に精錬を行う．iZ3を用い，偽反例か実反例かを判断し，偽反例の場合を精練述語を導出する．これにより，エラー状態に到達するたび，自動で反例解析及び精練述語の導出を行えるため，抽象遷移関係の計算を計算するコストがかからず，大幅な検証性能の向上が得ることができる．

2.5　検証実験

本研究は1章と同じマイコンH8/3687を基づいてに検証対象プログラムを用いて，検証実験を行う．

実行時間はアセンブリ命令の実行ステート数から求める．今回の検証対象プログラムは27行のアセンブリコードである．

プログラム（sample1）は，レジスタR0の絶対値を計算するプログラムである．

抽象化を用いてモデル検査を行う場合，エラーに到達することがあり，パスを精錬することで，エラー状態に到達しないこととリアルタイム性の両方を検証する．検証性質は以下のRTCTL7のように，すべてのパスにおいて，時間制約k以内にエラー状態に到達しない．

$$AG^{\leq k}\neg error \qquad (RTCTL\ 7)$$

本実験では，提案手法の有効性を検証するため．提案手法を含め，3つの手法で比較実験を行う．そして簡単化のために各命令及び制約時間kの単位は時間ではなく実行ステート数とし，それぞれ1及びk = 20とする．

- メソッド1は提案手法.
- メソッド2は従来のLazy Abstractionを用いて，Interpolationを用いてないとなる．時間付き到達木の精錬は最弱事前条件により精錬行う．
- メソッド3はLazy Abstractionを用いていない，従来の抽象化手法　Predicate Abstractionを用いた手法となる．

そして実験結果は表2.5.1で示す．提案手法が比較的効率的リアルタイム性検証を行えるを示している．提案手法の有効性を実証した．

表 2.5.1: The results of comparison of the three methods

Method	States	Relations	Counterexample analysis time(ms)	Verification time(ms)	Result
1	17	18	112	146	T
2	17	18	207	229	T
3	17	18	634	661	T

2.6　おわりに

本論文では，状態の回避と検証の高速化を目的とし，アセンブリプログラムを対象とし、リアルタイム性検証のための時間付きモデルをLazy Abstractionによる抽象化とInterpolationを用いた精練による状態削減手法を提案し、実験を行うことで有効性を示した．

今後の課題は検証におけるより複雑な検証対象を実証すべきである．現在のモデルに対しては検証できない性質がある．特に2章の検証システムはより複雑な検証対象に対して検証を行うためには量化子付きモデルを考慮する必要がある．

令和　3年　1月27日

学位論文審査報告書（甲）

1．学位論文題目（外国語の場合は和訳を付けること。）

モデル検査手法を用いた組込みアセンブリプログラムのリアルタイム性の形式的検証

(Formal Verification of Real-Time Properties for Embedded Assembly Program Using Model Checking)

2．論文提出者　(1) 所　　属　電子情報科学　　専攻

(2) 氏（ふり）　名（がな）　呉　亜軍（ご　あぐん）

3．審査結果の要旨（600～650 字）

当該学位論文に関して，令和３年１月２７日に第１回学位論文審査委員会を開催した．同日に口頭発表を実施し，その後に第２回審査委員会を開催した．慎重審議の結果，以下の通り判定した．なお，口頭発表における質疑を最終試験に代えるものとした．

本論文は，組込みアセンブリプログラムのリアルタイム性のモデル検査に関する研究をまとめたものであり，以下の２つの提案からなる．(1) リアルタイム性をモデル検査するために，アセンブリプログラムの実行時間を含めたリアルタイム時相論理の不動点計算手法を開発して，モデル検査のアルゴリズムを提案している．(2) 現実のアセンブリプログラムをモデル検査するために，数理論理学に基づく述語抽象化とその精錬の手法を開発して，定理証明技術によるモデル検査手法を提案している．以上の (1) と (2) の提案によるモデル検査のアルゴリズムと手法を実装して，計算機実験により，従来不可能であったアセンブリプログラムのリアルタイム性の効率的な形式的検証が実現でき，その有効性を実証した．

以上のように，組込みシステムの形式的検証において，組込みアセンブリプログラムのリアルタイム性の効率的なモデル検査手法を提案し実証したことは，学術上の重要な成果である．よって，本論文は博士（工学）に値すると判定した．

4．審査結果　(1) 判　　定（いずれかに○印）　(合　格)　・　不合格

(2) 授与学位　　博　士（　工学　）

5．学位論文及び参考論文に不適切な引用や剽窃が無いことの確認

☑　確認済み（確認方法：　iThenticate による　　）

☐　未 確 認（理　　由：　　）

6．学位論文審査委員

所　属（専攻名）		職　名	氏　名	
電子情報科学専攻	（主査）	教授	山根　智	印
電子情報科学専攻	（副査）	教授	藤解　和也	印
電子情報科学専攻	（副査）	准教授	深山　正幸	印
電子情報科学専攻	（副査）	准教授	今村　幸祐	印
電子情報科学専攻	（副査）	准教授	軸屋　一郎	印

令和　3年　1月27日

学位論文審査報告書（甲）

1．学位論文題目（外国語の場合は和訳を付けること。）

モデル検査手法を用いた組込みアセンブリプログラムのリアルタイム性の形式的検証

(Formal Verification of Real-Time Properties for Embedded Assembly Program Using Model Checking)

2．論文提出者　(1) 所　　属　電子情報科学　　専攻

(2) 氏（ふり）　名（がな）　呉　亜軍（ご　あぐん）

3．審査結果の要旨（600～650字）

当該学位論文に関して，令和3年1月27日に第1回学位論文審査委員会を開催した．同日に口頭発表を実施し，その後に第2回審査委員会を開催した．慎重審議の結果，以下の通り判定した．なお，口頭発表における質疑を最終試験に代えるものとした．

本論文は，組込みアセンブリプログラムのリアルタイム性のモデル検査に関する研究をまとめたものであり，以下の2つの提案からなる．(1) リアルタイム性をモデル検査するために，アセンブリプログラムの実行時間を含めたリアルタイム時相論理の不動点計算手法を開発して，モデル検査のアルゴリズムを提案している．(2) 現実のアセンブリプログラムをモデル検査するために，数理論理学に基づく述語抽象化とその精錬の手法を開発して，定理証明技術によるモデル検査手法を提案している．以上の (1) と (2) の提案によるモデル検査のアルゴリズムと手法を実装して，計算機実験により，従来不可能であったアセンブリプログラムのリアルタイム性の効率的な形式的検証が実現でき，その有効性を実証した．

以上のように，組込みシステムの形式的検証において，組込みアセンブリプログラムのリアルタイム性の効率的なモデル検査手法を提案し実証したことは，学術上の重要な成果である．よって，本論文は博士（工学）に値すると判定した．

4．審査結果　(1) 判　　定（いずれかに○印）　(合　格)　・　不合格

(2) 授与学位　　博　士（　工学　）