ソフトウェア工学 107－14
（1996．1．19）

オートマトン理論に基づくリアルタイムソフトウェアの検証システム

山根　智*　　蒋　海雲**
*島根大学総合理工学部数理情報システム学科
**南京通信研究所（中華人民共和国）

リアルタイムソフトウェアはプロセス構成が複雑でタイミング制約が厳しいために，タイミング制約を含む形式的検証に関する多くの研究が行われている．そこで，我々は時間ステートチャートによる仕様記述と検証の手法について報告した．本手法では，言語包合アルゴリズムにより公平性や正則性の検証を実現したが，検証コストが大きすぎる問題点がある．その原因としては，初期状態からの受理ループ探索や状態遷移関係をリスト構造で表現したことによる．本論文では，検証コストを削減するために，以下の効率的なタイミング検証手法を提案する．
(1)タイミング制約の充足性を判定しながら，受理条件を充足する強連結成分を探索して，強連結成分が初期状態から到達可能かどうかを判定することにより，計算コストを削減する．
(2)状態遷移関係を二分決定グラフで表現して，タイミング制約の充足性を判定しながら，像計算することにより，計算コストとメモリコストを削減する．
本論文では，上記手法を支援するタイミング検証システムを実現して，その有効性を確認した．

Verification system for real-time software based on automaton theory

Satoshi Yamane*　　Kaiun Chan**

* Dept. of computer science　Shimane University
**Nankin telecomunication laboratory in China

As real-time software consists of complex configurations of processes and has strict timing constraints, there have been many researches about formal verification including timing verification. We have reported the specification and verification method by timed statechart. Using this method, we have verified the properties of fairness and regularity by language inclusion algorithm, but the verification costs are large. In this paper, we propose the effective timing verification method as follows.
(1)We verify real-time software both by the finite set of strongly connected components of specification and by checking timing constraints.
(2)We verify real-time software both by image computation based on BDD(Binary Decision Diagram) and by checking timing constraints.
We have developed the verification system based on this method. This system can avoid state explosion problem.

1．まえがき

リアルタイムソフトウェアはプロセス構成が複雑でタイミング制約が厳しい．このために，システム仕様が検証性質を充足するかどうかを保証するための形式的検証の研究は重要である．以前，我々は時間ステートチャートに基づく検証の手法について報告した[1]．本手法では，言語包合アルゴリズムにより公平性や正則性の検証を実現したが，検証コストが大き過ぎる問題点がある．その原因としては，初期状態からの受理ループ探索や状態遷移関係をリスト構造で表現したことによる．本論文では，検証コストを削減するために，強連結成分の探索に基づくタイミング検証手法を提案する．

(1)状態遷移関係をリスト構造で表現する．タイミング制約の充足性を判定しながら，受理条件を充足する強連結成分[2]を探索して，強連結成分が初期状態から到達可能かどうかを判定することにより，計算コストを削減する．

(2)状態遷移関係を二分決定グラフ(Binary Decision Diagram(BDD))[3]で表現する．タイミング制約の充足性を判定しながら，像計算することにより，メモリコストを削減する．

本論文では，上記手法を支援するタイミング検証システムを実現して，実用レベルに近い事例によりその有効性を確認した．

以下の本論文は，２章では仕様記述と検証の手法を定義する．３章では強連結成分による検証手法を提案して，４章では二分決定グラフによる検証手法を提案する．５章では検証システムと検証事例を述べて，６章ではまとめを述べる．

2．仕様記述と検証の手法

2．1　仕様記述手法

システム仕様と検証仕様の記述言語としては，時間Mullerオートマトン[4]を拡張したプロセス時間オートマトンを使用する．個々のプロセスはプロセス時間オートマトンで記述して，システム仕様はプロセス時間オートマトンのカルテジアン積で定義する．

［定義１］（プロセス時間オートマトン）

プロセス時間オートマトンは（Σ，S，s０，C，E，Inv，F）の７つ組で定義される．

ここで，

Σ：有限イベント集合

S：有限状態集合

$s0 \subseteq S$：初期状態集合

C：クロックの有限集合

$E \subseteq S \times S \times \Sigma \times 2^{C} \times \Phi(C)$：状態遷移関数

$Inv \in S \rightarrow \Phi(C)$：Sで成立するタイミング制約

$F \subseteq 2^{S}$：受理状態集合のクラス

なお，$|s0| = 1$であり，状態遷移は決定的である．ここで，$\lambda \subseteq C$は状態遷移に伴ってリセットされるクロックを与える．Φ（C）はクロックCのタイミング制約式δでありクロック変数の集合X（∋x）と整数の時刻定数dにより再帰的に定義される．

$\delta ::= X \leqq d \mid d \leqq X \mid \neg\delta \mid \delta_1 \wedge \delta_2$

また，所与のイベントにより，無限回遷移する状態の集合がFに属すれば，そのイベントは受理される．■

2．2　検証手法

検証問題はシステム仕様の受理言語が検証性質仕様の受理言語に包合されるかどうかを判定するといった言語包合問題[4]に帰着する．

［定義２］（言語包合問題の定義）

言語包合問題は次のように定義できる．

$L(M1) \subseteq L(M2)$

また，これは以下と等価である．

$L(M1) \cap \overline{L(M2)} = \phi$

ここで，

M１：システム仕様の時間オートマトン

L（M１）：時間オートマトンM１の受理言語

M２：検証性質仕様の時間オートマトン

L（M２）：時間オートマトンM２の受理言語

上記の定義より，言語包合が成立するためには，L（M２）が補集合閉包性を持つ必要がある．さらに，L（M２）の補集合を言語として持つオートマトンの存在が必要となる．■

プロセス時間オートマトンは積演算閉包性や補集合演算閉包性がある．システム仕様と検証仕様の補集合との積時間オートマトンの生成は文献４）を拡張して容易に定義できる．

言語包合問題に基づく検証アルゴリズムは以下のとおりである[1]．

［定義３］（検証アルゴリズム）

検証アルゴリズムは以下の手続きで構成する．

(1)まず，検証仕様の補集合とシステム仕様との積時間オートマトンを生成する．

(2)Tarjanの深さ優先探索[5]により，積時間オートマトンの受理ループを発見する．

(3)もし，受理ループが存在すれば検証仕様は充足されない．そうでなければ検証仕様は充足される．■

3．リスト構造に基づく検証手法

文献１）の検証手法では，Tarjanの深さ優先探索[5]により，しらみつぶしに初期状態から積時間オートマトンの受理ループを発見していた．このために，探索時間のコストが大きく，実用的な問題の検証が不可能であった．そこで，本論文では，タイミング制約の充足性を判定しながら，受理条件を充足する強連結成分[2]を探索して，強連結成分が初期状態から到達可能か

どうかを判定することにより，タイミング検証する手法を提案する．

［定義４］（強連結成分に基づくタイミング検証手法）

```
プログラム開始；
まず，受理条件を充足する強連結成分を探索する．
if(強連結成分が存在しない)
 then 検証性質を充足すると判定する
 else if (強連結成分の状態列がタイミング制約を充足しない)
   then 検証性質を充足すると判定する
   else if(タイミング制約を充足しながら初期状態から強連結成分に到達する)
     then 検証性質を充足しないと判定する
     else 検証性質を充足すると判定する
プログラム終了；
```

■

［定義４］では，初期状態からの強連結成分への到達性判定は初期状態から深さ優先探索により容易に実現できる．しかし，受理条件を充足する強連結成分の探索やタイミング制約の充足性判定は多くの説明が必要なので，以下に説明する．

３．１　強連結成分の探索

受理条件を充足する強連結成分の探索アルゴリズムは以下のとおりである．深さ優先探索で各状態を探索し終わったときに，その状態の順序数（訪問の順番）が最小であれば，その状態およびその子孫の状態を強連結成分と見なして，受理条件を充足すれば強連結成分として出力する．

［定義５］（受理条件を充足する強連結成分の探索）

```
プログラム開始；
状態遷移関係をリスト構造で表現する；
各状態には順序数とフラグを存在させる；
    ／＊　順序数は訪問の順番を表わす　＊／
    ／＊　フラグは訪問済／未訪問を表わす　＊／
／＊　サブルーチン開始　＊／
サブルーチンvisit（ｖｉ）開始；
    ／＊　ｖｉ：パラメータ　＊／
作業変数ＷＯＲＫを０に初期化する；
順序数に１を加算する；
順序数の最小値に順序数を代入する；
ｖｉに順序数を代入する；
／＊　状態の探索開始　＊／
While(ｖｉから遷移する受理条件を構成する状態ｖｊが存在するか)
  if(ｖｊが未訪問か)
   then ＷＯＲＫ：＝サブルーチンvisit（ｖｊ）；
    ／＊　visitの返す値をＷＯＲＫに代入する　＊／
   else ＷＯＲＫ：＝訪問済の状態ｖｊの順序数；
  if ＷＯＲＫ＜順序数の最小値
    then 順序数の最小値にＷＯＲＫを代入する；
end；
／＊　状態の探索終了　＊／
／＊　強連結成分の出力開始　＊／
if(順序数の最小値とｖｉの順序数が等しいか)
 then 強連結成分を出力する；
／＊　強連結成分の出力終了　＊／
ＷＯＲＫを呼び出し元のサブルーチンvisitに返す；
サブルーチンvisit（ｖｉ）終了；
／＊　サブルーチン終了　＊／
順序数を０に初期化する；
順序数の最小値を０に初期化する；
  ／＊　順序数の最小値を保存する変数　＊／
各状態に存在するフラグを初期化する；
for(受理条件を構成する状態ｖｉ(i=1,,,n)毎に繰り返す)
  サブルーチンvisit（ｖｉ）を呼び出す；
プログラム終了；
```

■

３．２　タイミング制約の充足性判定

次に，強連結成分の状態列のタイミング制約の充足性判定及び初期状態から強連結成分への状態列のタイミング制約の充足性判定について説明する．検証コストを削減するために，従来の時間オートマトンの検証で使われていたリージョングラフ[4]を生成しない．タイミング制約は各状態毎にDBM(Difference Bounds Matrices)と呼ばれる時間不等式の行列で保持して，タイミング制約の充足性を判定する[6]．

［定義６］（タイミング制約の充足性の判定）

タイミング制約の充足性の判定の手続きは以下の処理から構成する．

(1)状態のタイミング制約より，各状態毎のDBMを生成する．

(2)Floyd-Warshallのアルゴリズムにより，DBMから正規形DBMを計算する．

(3)状態遷移元と状態遷移先の正規形DBMが交わりを持つかどうかを判定することにより，タイミング制約の充足性を判定する．■

以下では，(1)～(3)を詳細に定義する．

［定義７］（DBM(Difference Bounds Matrices)）

DBMはクロック制約割り当てのマトリックスである．クロック制約割り当ての形式は以下のとおりである．

$$t_i - t_j \leqq d_{ij}$$

ここで，

t_i，t_j：クロック変数

d_{ij}：クロック定数

DBMの（ｉ，ｊ）成分はd_{ij}である．d_{ij}は集合$\{.....,-2, -1, 0, 1, 2,.....\} \cup \{.....,-2^-, -1^-, 0^-, 1^-, 2^-,.....\} \cup \{-\infty, \infty\}$の要素である．例えば，$2^-$は$1<2^-<2$を充足する限りなく2に近い数である．これにより，$t_i - t_j < d_{ij}$は$t_i - t_j \leqq d_{ij}^-$と表現する．なお，仮想クロック$t_0 = 0$と定義する．■

［定義８］（正規形DBM）

DBMの各クロック変数をノードと見なして，各クロッ

ク定数をエッジのコストと見なすことにより，DBMがグラフ表現できる．これにより，Floyd-Warshallのアルゴリズムにより冗長のない最短パスである正規形DBMを求めることができる．■

［定義９］（タイミング制約の充足性の判定）
状態遷移元と状態遷移先の正規形DBMのintersectionを生成して，充足性を判定する．
(1)状態遷移系列において，状態遷移元と状態遷移先の正規形DBMのintersectionを生成する．

intersection DBM＝min{ d_{ij}, d'_{ij} }

ここで，

［d_{ij}］：状態遷移元の正規形DBM

［d'_{ij}］：状態遷移先の正規形DBM

(2)intersection DBMの正規形の閉ループに負のコストが存在すればタイミング制約を充足しない．そうでなければ，タイミング制約を充足する．■

［定理１］（タイミング制約の充足性判定の正当性）
intersection DBMの正規形の閉ループに負のコストが存在すれば到達可能性を充足しない．

（証明の方針）
クロック変数とクロック定数をグラフのノードとコストで表現することにより証明できる．■

３．３　タイミング検証の事例

最後に，強連結成分に基づく検証の事例を示す．

［例１］（強連結成分に基づくタイミング検証の事例）
図１で与えられたシステム仕様と検証仕様に対して，タイミング検証の事例を示す．仕様の受理状態集合のクラスを｛｛ｓ０，ｓ１，ｓ２，ｓ３，ｓ４｝｝，検証仕様の受理状態集合のクラスを｛｛ｖ０，ｖ１｝｝とする．検証仕様の補集合の受理状態集合のクラスは｛｛ｖ０｝，｛ｖ１｝｝となり，積時間オートマトンの受理状態集合のクラスは｛｛ｓ０×ｖ０，ｓ１×ｖ０，ｓ２×ｖ０，ｓ３×ｖ０，ｓ４×ｖ０｝，｛ｓ０×ｖ１，ｓ１×ｖ１，ｓ２×ｖ１，ｓ３×ｖ１，ｓ４×ｖ１｝｝となる．

図２(1)に示す積時間オートマトンの強連結成分を探索すると，｛ｓ０×ｖ０，ｓ１×ｖ０，ｓ３×ｖ０，ｓ４×ｖ０，ｓ２×ｖ０｝と｛ｓ０×ｖ１，ｓ１×ｖ１，ｓ３×ｖ１，ｓ４×ｖ１，ｓ２×ｖ１｝という２つの強連結成分が探索できた．強連結成分のすべての状態遷移関係のタイミング制約の充足性を判定する必要がある．ここでは，強連結成分の状態遷移ｓ０×ｖ１→ｓ１×ｖ１→ｓ３×ｖ１→ｓ４×ｖ１→ｓ２×ｖ１のタイミング制約の充足性判定を示す．まず，各状態のDBMを生成する．例えば，状態遷移ｓ０×ｖ１→ｓ１×ｖ１のタイミング制約の充足性判定を示す．

ｓ１×ｖ１では

t≦7∧u≦8よりt - t0≦7，u - t0≦8

と変形できるので，DBM D1が生成できて，Floyd-Warshallのアルゴリズムにより正規形DBMが計算できる．ｓ０×ｖ１のDBM D4も同様に計算できる．次に，ｓ０×ｖ１→ｓ１×ｖ１のタイミング制約の充足性判定をするために，intersection D0∩D1 DBMを生成して，Floyd-Warshallのアルゴリズムにより正規形intersection DBMが計算できる．この例では，正規形intersection DBMの閉ループに負のコストが存在しないので，状態遷移ｓ０×ｖ１→ｓ１×ｖ１は可能である．他の状態遷移のタイミング制約の充足性判定も同様に計算できる．

次に，深さ優先探索により，初期状態から強連結成分への到達可能性をチェックすると，２つの強連結成分はどちらも初期状態から到達可能であり，タイミング制約の充足性も満たされた．

以上より，この仕様は検証仕様を充足しないことが検証できた．■

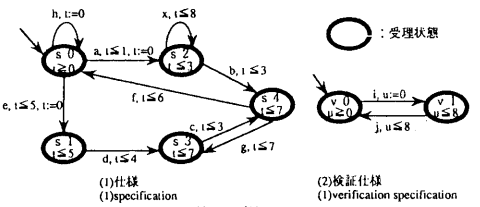
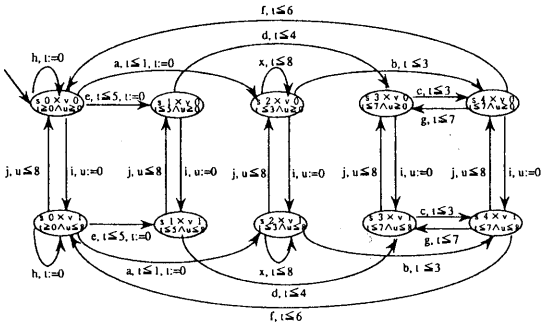


図１　検証の事例
Fig.1　Example of verification



At state s0v1, from t≧0∧u≦8
DBM D0=0 * *　正規形 DBM=0 * *
8 * *　8 * *
At state s1v1, from t≦5∧u≦8
DBM D1=5 * *　正規形 DBM=5 * *
8 * *　8 * *
At state s2v1, from t≦3∧u≦8
DBM D2=3 * *　正規形 DBM=3 0 0
8 * *　8 0 0
At state s3v1, from t≦7∧u≦8
DBM D3=7 * *　正規形 DBM=7 * *
8 * *　8 * *
At state s4v1, from t≦7∧u≦8
DBM D4=7 * *　正規形 DBM=7 * *
8 * *　8 * *

At state s0v1→states1v1, from t≧0∧u≦8 to t≦5∧u≦8.
intersection D0∩D1 DBM=5 * *　正規形 DBM=5 * *
8 * *　8 * *

(2)タイミング制約の充足性判定
(2)Testing for timing constraints

図２　タイミング制約の充足性判定の事例
Fig.2　Example of testing for timing constraints

4．二分決定グラフに基づく検証手法

二分決定グラフ（以下ではBDDと記す）[3]は状態遷移図をコンパクトに表現できるために，シンボリックモデルチェッキングなどの形式的検証に応用されている．本論文では，このBDDを使って，タイミング制約の充足性を判定しながら，受理条件を充足する強連結成分を探索して，強連結成分が初期状態から到達可能かどうかを判定するといったタイミング検証手法を提案する．

積時間オートマトンにはタイミング制約が状態遷移関係に存在するために，従来のBDDを使った形式的検証手法[7]では積時間オートマトンを検証できない．積時間オートマトンをBDDを使って形式的検証するためには，以下の２つの方法が考えられる．

(1)積時間オートマトンからタイミング制約のないリージョングラフを生成する．リージョングラフ[4]は従来の手法でBDDにエンコードできて，強連結成分の探索などがBDDで実現できる．しかし，リージョングラフはタイミング制約の定数の長さの指数オーダで状態遷移関係が増大するので，状態遷移関係をBDDにエンコードしても検証コストの削減はあまり期待できないと考えられる．

(2)積時間オートマトンの状態遷移関係をBDDにエンコードして，各状態のタイミング制約はDBMで表現する．すなわち，積時間オートマトンの各状態は，状態ｓｉとタイミング制約が構成する時間領域ｚｉの構造体（ｓｉ，ｚｉ）（ｉ＝１,...,ｎ）で保持する．ここで，ｓｉはBDDでありｚｉはDBMである．探索手法としては以下のとおりである．

(a)BDDにエンコードされた状態遷移関係Ｒと状態ｓｉとの論理関数演算により，状態遷移する状態集合を計算する．

(b)計算した状態集合を各状態ｓｊ（ｊ＝１,...,m）に分解して，構造体（ｓｉ，ｚｉ）の集合を参照して，各状態ｓｊと状態ｓｉとのタイミング制約の充足性を判定する．

この方法は，従来のBDDを使った形式的検証手法の利点を生かしながら，タイミング制約の充足性の判定が可能である．

本論文では，以上の考察より，(2)の方法で積時間オートマトンの強連結成分の探索などを実現する．

［定義１０］（二分決定グラフに基づく検証手法）

二分決定グラフに基づくタイミング検証手法は以下の手続きから構成する．

(1)まず，積時間オートマトンの各状態を構造体（ｓｉ，ｚｉ）（ｉ＝１,....,ｎ）で表現して，状態遷移関係ＲをBDDにエンコードする．ここで，ｓｉはBDD，ｚｉはDBM，ｎは状態数である．

(2)次に，初期状態から到達可能な状態集合を計算する．

(3)次に，初期状態から到達可能な状態集合に制限した状態集合の推移的閉包を計算する．

(4)最後に，推移的閉包から受理条件を充足する強連結成分を計算する．

(a)もし，強連結成分が存在しなければ検証性質を充足すると判定する．

(b)もし，強連結成分が存在してタイミング制約を充足すれば，検証性質を充足しないと判定する．■

4．1　状態遷移関係のBDDへのエンコード

まず，論理ベクタ｛０，１｝^{n}の集合Ｅの部分集合Ａに対して，次の条件を満たす関数χ_A：Ｅ→｛０，１｝はＡの特性関数として定義される．

$$\chi_A(\overline{x}) = 1 \Leftrightarrow \overline{x} \in A$$

今，ベクタ関数$\tau = \lambda(v_1, \cdots, v_n).f$を定義する．ここで，

$v_i \in \{0, 1\}$

τ：特性関数

ｆ：変数$v_1, \cdots, v_n$をとる関数

ゆえに，任意の状態の集合Ｓに対して，

$$S = \lambda(v_1, \cdots, v_n).\overline{S}$$

を満たす論理関数表現$\overline{S}$が定義できる．以上より，構造体（ｓｉ，ｚｉ）のｓｉがBDDにエンコードできる．

［定義１１］（構造体（ｓｉ，ｚｉ）の定義）

任意の状態ｓｉに対して，構造体（ｓｉ，ｚｉ）（ｉ＝１,...,ｎ）が以下のように定義できる．

(1)ｓｉについて

任意の状態ｓｉに対して，

$$s_i = \lambda(v_1, \cdots, v_n).\overline{s}_i$$

を満たす論理関数表現$\overline{s}_i$が定義できる．

(2)ｚｉについて

［定義７］により，任意の状態の集合ｓｉに対して，DBMが定義できる．

以上より，構造体（ｓｉ，ｚｉ）（ｉ＝１,...,ｎ）が定義できる．■

次に，ベクタペア関数

$\tau = \lambda((v_1, \cdots, v_n), (v_1', \cdots, v_n')).f$

を考える．これにより，状態遷移関係Ｒに対して，BDD表現が定義できる．

［定義１２］（状態遷移関係のBDD表現）

状態遷移関係ＲのBDD表現は，

$R = \lambda((v_1, \cdots, v_n), (v_1', \cdots, v_n')).\overline{R}$

を満たす論理関数表現$\overline{R}$で定義できる．■

4．2　初期状態から到達可能な状態集合

初期状態から到達可能な状態集合の計算方法は，状態遷移関係のBDD表現を使って，論理関数演算による像計算(image computation)で定義できる．

［定義１３］（初期状態から到達可能な状態集合）

プログラム開始；

$\overline{S}(\overline{v}) := \overline{S}1(\overline{v})$；

repeat {
$S(\overline{v'}) := \exists \overline{v}.[R(\overline{v}, \overline{v'}) \wedge S(\overline{v})]$;
if $(S(\overline{v'}) \Rightarrow S(\overline{v})) = 1$ then 終了;
$S(\overline{v}) := S(\overline{v'}) \vee S(\overline{v})$;
}
プログラム終了;
ここで,
$R(\overline{v}, \overline{v'})$ ：状態遷移関係のBDD表現
$S(\overline{v'})$ ：状態遷移先の状態集合のBDD表現
$S(\overline{v})$ ：状態遷移元の状態集合のBDD表現
$S1(\overline{v})$ ：初期状態集合のBDD表現 ■

4．3 推移的閉包の計算

状態遷移関係に対する推移的閉包を計算するアルゴリズムとしては，反復法や反復二乗法，再帰法[8]などがある．本論文では，最も簡単な反復法[2, 9]を利用して推移的閉包を計算する．

［定義１４］（推移的閉包の計算アルゴリズム）
プログラム開始;
／＊初期状態から到達可能な状態集合に制限する＊／
$R(\overline{v}, \overline{v'}) := R(\overline{v}, \overline{v'}) \wedge S(\overline{v})$;
／＊推移的閉包の計算＊／
repeat {
$T(\overline{v}, \overline{v'})$
$:= \exists \overline{u}.[R(\overline{v}, \overline{u}) \wedge R(\overline{u}, \overline{v'})] \vee R(\overline{v}, \overline{v'})$;
if $(T(\overline{v}, \overline{v'}) = R(\overline{v}, \overline{v'}))$ then 終了;
$R(\overline{v}, \overline{v'}) := T(\overline{v}, \overline{v'})$;
}
プログラム終了;
ここで,
$R(\overline{v}, \overline{v'})$ ：状態遷移関係のBDD表現 ,
$S(\overline{v})$ ：初期状態から到達可能な状態集合のBDD表現
$T(\overline{v}, \overline{v'})$ ：初期状態から到達可能な状態集合に制限した推移的閉包のBDD表現 ■

4．4 強連結成分の計算

強連結成分を計算するアルゴリズムは，推移的閉包を用いて容易に実現できる．

［定義１５］（強連結成分の計算アルゴリズム）
プログラム開始;
$SCC(\overline{v})$
$:= \exists \overline{v'}.[T(\overline{v}, \overline{v'}) \wedge T(\overline{v'}, \overline{v})] \wedge A(\overline{v'})$;
if （$SCC(\overline{v})$ が受理条件を充足する強連結成分を含まない）
then 検証性質が充足される；
else if (強連結成分がタイミング制約を充足するか)
then 検証性質が充足されない；
else 検証性質が充足される；
プログラム終了;
ここで,
$SCC(\overline{v})$ ：強連結成分の状態集合のBDD表現
$T(\overline{v}, \overline{v'})$ ：推移的閉包のBDD表現
$T(\overline{v'}, \overline{v})$ ：推移的閉包のBDD表現
$A(\overline{v'})$ ：受理条件を構成する受理状態集合のクラスの状態集合のBDD表現 ■

4．5 タイミング検証の事例

二分決定グラフに基づく検証の事例を以下に示す．

［例２］（二分決定グラフに基づく検証事例）
前述の［例１］の事例を使用して，二分決定グラフに基づくタイミング検証を説明する．まず，仕様と検証仕様の補集合の時間オートマトンとの積時間オートマトンの状態遷移関係をBDDにエンコードする．また，積時間オートマトンの各状態を構造体（ｓｉ，ｚｉ）（ｉ＝１,...,ｎ）で表現する．これらを図３に示す．

次に，初期状態から到達可能な状態集合を計算する．本事例では，すべての状態が初期状態から到達可能であった．次に，初期状態から到達可能な状態集合に制限した状態集合の推移的閉包を計算する．最後に，推移的閉包から受理条件を充足する強連結成分を計算する．この事例では，強連結成分が存在するので検証性質を充足しないことが判定できた．■

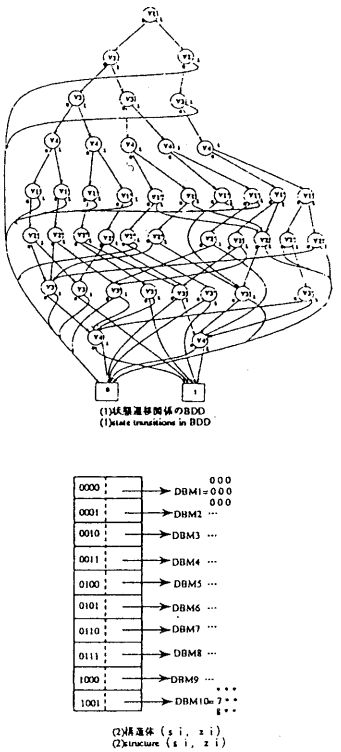


図３ 状態遷移関係のBDDと構造体（ｓｉ，ｚｉ）
Fig.3 state transitions in BDD and structure（ｓｉ，ｚｉ）

5．検証システムと検証事例

5．1　検証システム

本論文で提案した手法を支援する検証システムを実現した．すなわち，強連結成分の探索に基づいた，リスト構造及び二分決定グラフの検証システムを実現した．検証システムは図4に示すように，リスト構造の検証システムでは積時間オートマトン生成器や言語包含検証器から構成して，二分決定グラフの検証システムでは積時間オートマトンのBDD生成器やシンボリック言語包含検証器から構成した．これらの検証システムはC言語で実現しており，各々約1.5kstepと約3kstepである．仕様と検証仕様はプログラミング言語形式[6]で記述して生成器に入力する．生成器は仕様と検証仕様からリスト構造やBDD形式の積時間オートマトンなどを出力する．また，検証器は仕様が検証仕様を充足するかどうかを言語包含アルゴリズムにより判定する．なお，BDDに関する検証システムの実現は京都大学工学部情報工学科矢島研究室作成のSBDDライブラリ[10]を使用させて頂いた．

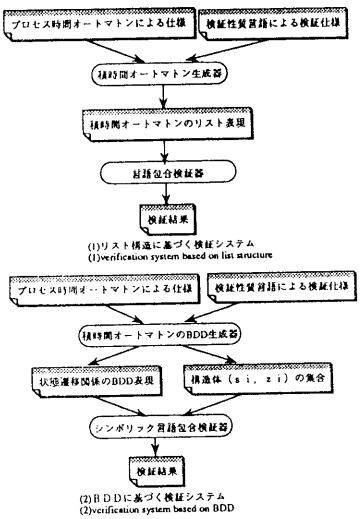


図4　検証システムの構成
Fig.4　Configuration of verification system

5．2　検証事例

本論文では，イーサネットのCSMA/CD[11]により提案した検証手法と検証システムの有効性を示す．

(1)問題の概要

イーサネットのCSMA/CDはLANで広く使われており，以下の特徴を有する．送信局はデータを送信すると，チャネルの応答を感知する．もし，チャネルがアイドルならば送信局はデータを送信する．しかし，チャネルがbusyであったりデータが破壊したら，ある時間待って再送する．以上のイーサネットのCSMA/CDプロトコルは図5のように仕様記述できる．検証性質記述は図6に示す．(1)と(2)は各々true/falseのタイミング制約に関する検証性質記述である．

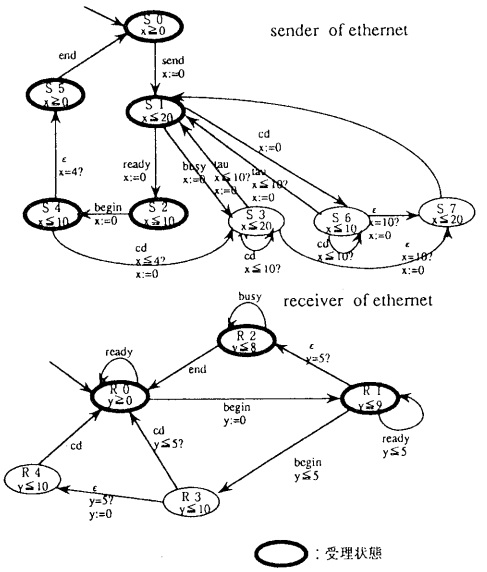


図5　イーサネットの仕様記述
Fig.5　specification of ethernet

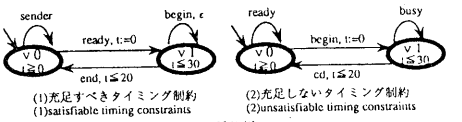


図6　検証性質仕様
Fig.6　verification specification

(2)検証

SUN4/IP（メモリ12MB)に実装した検証システムにより，本論文で提案した手法と従来の深さ優先探索やリージョングラフによる手法を比較評価した．senderとreceiverを増加させた仕様に対して，図6の検証性質仕様を検証して比較評価した．リージョングラフ手法と深さ優先探索手法では状態数数百で計算が止まらなかったりメモリ不足を起こした．今回提案した検証システムの検証コストの実測結果の一部を表1に示す．また，本論文で提案した2手法の所要メモリ量を比較すると，図7のようになり，二分決定グラフによる手法を使わないと大規模システムの検証が不可能なことがわかる．

表1　検証のメモリ量と計算時間の評価（一部）
Table1　Evaluation of space and time complexity(a part)

仕様		結果	強連結成分による検証		BDDによる検証	
状態数	遷移数		計算時間	メモリ量	計算時間	メモリ量
16	54	true	0.2sec	0.4Mb	0.9sec	2.8Mb
16	51	false	0.2sec	0.4Mb	1.1sec	2.8Mb
80	329	true	0.5sec	0.7Mb	6.7sec	2.8Mb
80	315	false	0.5sec	0.7Mb	9.2sec	2.9Mb
350	1926	true	3.4sec	2.4Mb	73.1sec	2.8Mb
350	1845	false	3.7sec	2.5Mb	111.2sec	2.9Mb
560	3141	true	7.9sec	3.9Mb	32.9sec	2.8Mb
560	3037	false	8.7sec	3.9Mb	400.8sec	2.9Mb
640	3710	true	9.9sec	4.3Mb	307.5sec	2.8Mb
640	3592	false	11.0sec	4.4Mb	447.5sec	2.9Mb
1024	6840	true	25.0sec	4.5Mb	532.6sec	2.8Mb
1024	6704	false	26.3sec	4.5Mb	587.2sec	2.9Mb

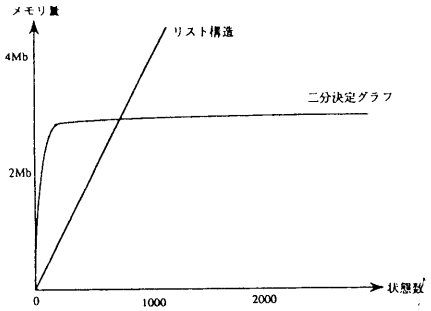


図7　メモリ量－状態数の関係
Fig.7　the relation between memory and states

6．むすび

本論文では，効率的な言語包合問題によるタイミング検証方式を提案して，その検証システムを実現して有効性を示した．提案したタイミング検証方式は，タイミング制約の充足性を判定しながら，リスト構造や二分決定グラフで表現した状態遷移関係を基礎として，受理条件を充足する強連結成分を探索するものである．本手法は従来の検証手法よりも数十倍効率的なことが実証できた．

今後の研究課題としては以下が考えられる．

(1)現在実現している検証システムは検証性質の充足性のみを判定する機能のみであり，バグ対処への支援機能がない．今後はバグ対処への支援機能などを含めたトータルな支援環境を提供する．

(2)現在実現している検証システムへの入力情報はプログラミング言語形式であり，複雑なオートマトンの仕様記述には不適当である．今後は可視された仕様記述手段を提供する．

参考文献

1)山根：" 時間ステートチャートに基づくリアルタイムシステム検証方式" ，情報処理学会論文誌，Vol.35, No.12,pp.2640-2650(1994)

2)Aho A.V.Hopcroft J.E. Ullman J.D.:The design and analysis of computer algorithms,Addison-Wesley(1974)

3)Bryant R.E.:"Graph-Based Algorithms for Boolean Function Manipulation",IEEE Transactions on Computers, Vol.C-35,No.8,pp.677-691(1986)

4)Alur R. Dill D. :"The Theory of Timed Automata", LNCS 600,pp.45-73(1992)

5)Tarjan R.,"Depth-first search and linear graph algorithms",SIAM J. of Computing1(2),pp.146-160(1972)

6)山根：" 実時間並行ソフトウェアの仕様記述と検証" ,情報処理学会論文誌,Vol.37,No.2(1996)(to appear)

7)McMillan K.L.:Symbolic Model Checking,Kluwer, P.194(1993)

8)松永，他：" 2分決定グラフを用いた推移的閉包計算アルゴリズムと形式的検証への応用" ,情処学研報DA65-7,pp.49-56(1993)

9)Touatic H. et-al.:"Testing language containment for ω-automata using BDDs",Info. and comp.118,pp.101-109 (1995)

10)Minato S. et-al.:"Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation",Proc. 27th DAC,pp.52-57(1990)

11)IEEE ANSI/IEEE 802.3, ISO/DIS 8802/3. IEEE CS (1985)