(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-5674
(P2004-5674A)
(43)公開日　平成16年1月8日(2004.1.8)

(51)Int.Cl.[7]	F I	テーマコード（参考）
G06F 17/50	G06F 17/50 664Z	5B046

審査請求 有　請求項の数 74　ＯＬ　（全 42 頁）

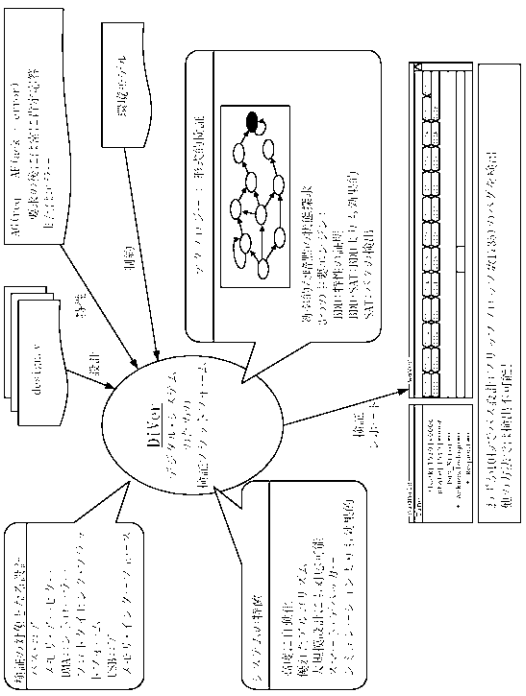
(21)出願番号　特願2003-141423(P2003-141423)
(22)出願日　平成15年5月20日(2003.5.20)
(31)優先権主張番号　10/157486
(32)優先日　平成14年5月30日(2002.5.30)
(33)優先権主張国　米国(US)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100093595
弁理士　松本　正夫
(72)発明者　マレイ　ガナイ
アメリカ合衆国、ニュージャージー　０８５４０　プリンストン、４　インディペンデンス　ウェイ　エヌ　イー　シー　ユー　エス　エー　インク内
(72)発明者　リンタオ　ジャン
アメリカ合衆国、ニュージャージー　０８５４０　プリンストン、４　インディペンデンス　ウェイ　エヌ　イー　シー　ユー　エス　エー　インク内

最終頁に続く

(54)【発明の名称】効率的な有界モデル検査方法

(57)【要約】
【課題】アプリケーションのメモリ所要量が半減され、より大規模な回路にスケーリングすることが可能になる有界モデル検査方法を提供する。
【解決手段】任意線形時間論理時相特性の有界モデル検査方法であって、Ｆはｅｖｅｎｔｕａｌｉｔｙ演算子、Ｇはｇｌｏｂａｌｌｙ演算子、Ｕはｕｎｔｉｌ演算子、Ｘはｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）に関連付けられた特性を、ブーリアン充足可能性検査から成る特性検査スキーマに変換するステップから成る。特性全体は、Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）の各演算子に関する特性検査スキーマの反復呼び出しと、原子命題およびブール演算子の標準的な処理で構成されるカスタマイズされた方法で検査される。
【選択図】　図１



【特許請求の範囲】
【請求項１】
任意線形時間論理時相特性の有界モデル検査方法であって、下記要件を備える。
Ｆはｅｖｅｎｔｕａｌｉｔｙ演算子、Ｇはｇｌｏｂａｌｌｙ演算子、Ｕはｕｎｔｉｌ演算子、Ｘはｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）に関連付けられた特性を、ブーリアン充足可能性検査から成る特性検査スキーマに変換し、
特性全体を、Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）の各演算子に関する特性検査スキーマの反復呼び出しと、原子命題およびブール演算子の標準的な処理で構成されるカスタマイズされた方法により検査する。
【請求項２】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、次に検査する演算子に関して選択肢が存在する場合には、探索の難度により決定される優先順位に従って選択を行う。
【請求項３】
請求項２に記載の任意線形時間論理時相特性の有界モデル検査方法であって、前記探索の難度は、原子命題、Ｘ演算子、Ｆ演算子、Ｕ演算子、Ｇ演算子の順によって高くする。
【請求項４】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、特性検査スキーマのサブセットが、対応するブーリアン充足可能性問題をサイズの小さい複数のブーリアン充足可能性下位問題に分割するために、対応する有界モデル検査問題のｋ番目のインスタンスに対して分割を実行する。
【請求項５】
請求項４に記載の任意線形時間論理時相特性の有界モデル検査方法であって、前記分割を演算子間にまたがって実行し、各演算子に対しては時間フレーム間と時間フレーム内の両方で実行する。
【請求項６】
請求項５に記載の任意線形時間論理時相特性の有界モデル検査方法であって、前記分割は可能な場合は常に、漸増的に関連付けられるブーリアン充足可能性下位問題に到達するように方向付けする。
【請求項７】
請求項５に記載の任意線形時間論理時相特性の有界モデル検査方法であって、複数のブール充足可能性下位問題にまたがる学習を使用する。
【請求項８】
請求項５に記載の任意線形時間論理時相特性の有界モデル検査方法であって、複数の関連する下位問題にまたがる学習を行うために、ブール充足可能性アルゴリズムの漸増的な公式化を使用する。
【請求項９】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、定数の伝搬に基づく回路単純化を使用する。
【請求項１０】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、構造同形の検出に基づく回路の単純化を使用する。
【請求項１１】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、ブーリアン充足可能性検査問題を、回路ベースとＣＮＦベースの充足可能性アルゴリズムを結合したハイブリッドＳＡＴソルバーを使用して解決する。
【請求項１２】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、Ｆ（ｐ）の特性検査スキーマは下記ステップを備える。

ａ）　時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始し、ここでｉ＝０は初期状態に対応する。
ｂ）　現在のパスのｉ番目の状態におけるｐの充足可能性を検査する。
ｃ）　充足可能な場合は、探索を成功として終了する。
ｄ）　充足不能な場合は、ｐはｉ番目の状態においては常に偽であることを学習し、この学習した知識を制約データベースに追加する。
ｅ）　ｉを所定の上限まで増大させながらステップｂ～ｄを繰り返し実行することによって探索を継続する。
ｆ）　所定の上限に到達すると、探索を不確定のまま終了する。
【請求項１３】
請求項１２に記載の任意線形時間論理時相特性の有界モデル検査方法であって、ステップｄ～ｅ間の完全性を検査するステップをさらに備える。
【請求項１４】
請求項１３に記載の任意線形時間論理時相特性の有界モデル検査方法であって、完全性を検査するステップはさらに下記ステップを備える。
（ｉ）　ｉ番目の状態を起点とする遷移がパス内の検査済みの状態を訪れないようにするための制約を追加し、制約データベースの充足可能性を検査する。
（ｉｉ）　充足不能な場合は、探索を失敗として終了する。
（ｉｉｉ）　充足不能でない場合は、探索を継続する。
（ｉｖ）　以前の状態がすべて検査されるまでステップ（ｉ）～（ｉｉｉ）を繰り返す。
【請求項１５】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、Ｇ（ｐ）の特性検査スキーマは下記ステップを備える。
ａ）　時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始し、ここでｉ＝０は初期状態に対応する。
ｂ）　ｐが現在のパスのｉ番目の状態において充足されることを保証するための制約をデータベースに追加し、充足可能性を検査する。
ｃ）　充足不能な場合は、探索を失敗として終了する。
ｄ）　充足可能な場合は、開始状態からｉ番目の状態までの間でｊ番目の状態が各々ループバック状態かどうか検査し、ループバック状態が検出されたら探索を成功として終了する。
ｅ）　開始状態より前のｊ番目の状態が各々ループバック状態かどうか検査し、ループバック状態が検出されたら探索を成功として終了する。
ｆ）　ｉを所定の上限まで増大させながらステップａ～ｅを繰り返し実行することによって探索を継続する。
ｇ）　所定の上限に到達すると、探索を不確定のまま終了する。
【請求項１６】
請求項１５に記載の任意線形時間論理時相特性の有界モデル検査方法であって、ステップｄは以下によって実行される。
（ｄｉ）　ｉ番目の状態からｊ番目の状態までの遷移の充足可能性を検査する。
（ｄｉｉ）　ステップｄｉの遷移が充足可能な場合は、探索を成功として終了する。
（ｄｉｉｉ）　ステップｄｉの遷移が充足不能な場合は、このような遷移が存在しないことを学習し、この知識を制約データベースに追加する。
（ｄｉｖ）　開始状態からｉ番目の状態までのすべての状態が検査されるまで（ｄｉ）～（ｄｉｉｉ）を繰り返すことによって探索を継続する。
【請求項１７】
請求項１５に記載の任意線形時間論理時相特性の有界モデル検査方法であって、ステップｅは以下によって実行される。
（ｅｉ）　ｊ番目の状態から開始状態までの各状態のＰの充足可能性を検査する。
（ｅｉｉ）　ステップｅｉで充足不能な場合は、ステップｅを中止してステップｆに進む

。
（ｅｉｉｉ）　ステップｅｉで充足可能な場合は、ｉ番目の状態からｊ番目の状態までの遷移の充足可能性を検査する。
（ｅｉｖ）　ステップｅｉｉｉの遷移が充足可能な場合は、探索を成功として終了する。
（ｅｖ）　ステップｅｉｉｉの遷移が充足不能な場合は、初期状態（ｉ＝０）から開始状態までのすべての状態が検査されるまでステップｅｉ～ｅｉｖを繰り返すことによって探索を継続する。
【請求項１８】
請求項１５に記載の任意線形時間論理時相特性の有界モデル検査方法であって、ステップｅ～ｆ間の完全性を検査するステップをさらに備える。
【請求項１９】
請求項１８に記載の任意線形時間論理時相特性の有界モデル検査方法であって、完全性を検査するステップはさらに下記ステップを備える。
（ｉ）　ｉ番目の状態を起点とする遷移がパス内の検査済みの状態を訪れないようにするための制約を追加し、制約データベースの充足可能性を検査する。
（ｉｉ）　ステップｉで充足不能な場合は、探索を失敗として終了する。
（ｉｉｉ）　ステップｉで充足可能な場合は、探索を継続する。
（ｉｖ）　以前の状態がすべて検査されるまでステップ（ｉ）～（ｉｉｉ）を繰り返す。
【請求項２０】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、Ｕ（ｐ，ｑ）　の特性検査スキーマは下記ステップを備える。
ａ）　時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始し、ここでｉ＝０は初期状態に対応する。
ｂ）　現在のパスのｉ番目の状態におけるｑの充足可能性を検査する。
ｃ）　充足可能な場合は、探索を成功として終了する。
ｄ）　充足不能な場合は、ｑはｉ番目の状態においては常に偽であることを学習し、この学習した知識を制約データベースに追加する。
ｅ）　ｐがｉ番目の状態において充足されることを保証するための制約をデータベースに追加し、充足可能性を検査する。
ｆ）　ステップｅで充足不能な場合は、探索を失敗として終了する。
ｇ）　ステップｅで充足可能な場合は、探索を継続する。
ｈ）　ｉを所定の上限まで増大させながらステップｂ～ｇを繰り返し実行することによって探索を継続する。
ｉ）　所定の上限に到達すると、探索を不確定のまま終了する。
【請求項２１】
請求項２０に記載の任意線形時間論理時相特性の有界モデル検査方法であって、ステップｇ～ｈ間の完全性を検査するステップをさらに備える。
【請求項２２】
請求項２１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、完全性を検査するステップはさらに下記ステップを備える。
（ｉ）　ｉ番目の状態を起点とする遷移がパス内の検査済みの状態を訪れないようにするための制約を追加し、制約データベースの充足可能性を検査する。
（ｉｉ）　ステップｉで充足不能な場合は、探索を失敗として終了する。
（ｉｉｉ）　ステップｉで充足可能な場合は、探索を継続する。および
（ｉｖ）　以前の状態がすべて検査されるまでステップ（ｉ）～（ｉｉｉ）を繰り返す。
【請求項２３】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、Ｘ（ｐ）の特性検査スキーマは下記ステップを備える。
ａ）　時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始し、ここでｉ＝０は初期状態に対応する。

ｂ）　さらにもう１つの時間フレームに遷移関係を展開し、時間フレームｉ＋１においてｐが真であるかどうかを検査する。
ｃ）　ｐが時間フレームｉ＋１において真であることが確認された場合は、成功として終了する。
ｄ）　ｐが時間フレームｉ＋１において偽であることが確認された場合は、失敗として終了する。
ｅ）　ｐの検査が時間フレームｉ＋１において不確定の場合は、不確定として終了する。
【請求項２４】
請求項１２に記載の任意線形時間論理時相特性の有界モデル検査方法であって、充足可能性検査のサブセットが単一の充足可能性検査として結合される。
【請求項２５】
請求項１５に記載の任意線形時間論理時相特性の有界モデル検査方法であって、充足可能性検査のサブセットが単一の充足可能性検査として結合される。
【請求項２６】
請求項２０に記載の任意線形時間論理時相特性の有界モデル検査方法であって、充足可能性検査のサブセットが単一の充足可能性検査として結合される。
【請求項２７】
請求項２３に記載の任意線形時間論理時相特性の有界モデル検査方法であって、充足可能性検査のサブセットが単一の充足可能性検査として結合される。
【請求項２８】
線形時間時相論理で表現された安全性特性の帰納証明の方法であって、下記ステップを備える。
ｋインスタンスの基本ケースを、与えられた特性の否定に関して検査するステップ、ｋインスタンスの帰納ステップを、与えられた特性に対応するモニター述語に関して検査するステップ、ｋ＝０から開始し、任意に指定された上限までそれを漸増させるステップを備え、前記基本ケースの検査はさらに下記ステップを備える。
否定された特性を、Ｆがｅｖｅｎｔｕａｌｉｔｙ演算子、Ｘがｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）およびＸ（ｐ）と関連づけられたブール充足可能性検査から成る特性検査スキーマに変換するステップと、初期状態を起点として、Ｆ（ｐ）、Ｘ（ｐ）演算子に関する特性検査スキーマの反復呼び出しと、原子命題およびブール演算子の標準的な処理により、否定された特性の証拠を探索するステップ。
証拠が検出されないことが証明された場合は、証明全体を成功として終了する。
ｋ時間フレーム内に証拠が検出された場合は、証明全体を失敗として終了する。
結果が不確定の場合は、帰納ステップの検査を継続する。
この帰納ステップの検査は、以下の特性検査スキーマから成る。
モニター述語が任意状態からｋ時間フレームにわたって有効で、かつｋ＋１番目の時間フレーム内では有効ではない状況を保証するための制約を追加する。
充足不能な場合は、証明全体を成功として終了する。
充足不能でない場合は、探索を継続する。
前記探索を、ｋを所定の上限まで漸増させながら基本ケースと帰納ステップを繰り返し実行することで継続する。
ｋが指定された上限に到達した場合は、証明は不確定となる。
【請求項２９】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、ｋ時間フレーム内の状態が一意であることを保証するための制約を追加し、それにより完全な証明プロシージャーを提供する。
【請求項３０】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、次に検査する演算子に関して選択肢が存在する場合には、探索の難度により決定される優先順位に従って選択を行う。

【請求項３１】
請求項３０に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、前記探索の難度を、原子命題、Ｘ演算子、Ｆ演算子の順に高める。
【請求項３２】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、特性検査スキーマのサブセットが、対応するブーリアン充足可能性問題をサイズの小さい複数のブーリアン充足可能性下位問題に分割するために、帰納下位問題のｋ番目のインスタンスに対して分割を実行する。
【請求項３３】
請求項３２に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、前記分割を演算子間にまたがって実行し、各演算子に対しては時間フレーム間と時間フレーム内の両方で実行する。
【請求項３４】
請求項３３に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、この分割は可能な場合は常に、漸増的に関連付けられるブーリアン充足可能性問題に到達するように方向付けする。
【請求項３５】
請求項３３に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、複数のブール充足可能性問題にまたがる学習を使用する。
【請求項３６】
請求項３３に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、複数の関連する下位問題にまたがる学習を行うために、ブール充足可能性アルゴリズムの漸増的な公式化を使用する。
【請求項３７】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、定数の伝搬に基づく回路単純化を使用する。
【請求項３８】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、構造同形の検出に基づく回路の単純化を使用する。
【請求項３９】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、ブーリアン充足可能性検査問題を、回路ベースとＣＮＦベースの充足可能性アルゴリズムを結合したハイブリッドＳＡＴソルバーを使用して解決する。
【請求項４０】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、Ｆ（ｐ）の特性検査スキーマは下記ステップを備える。
ａ）　時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始し、ここでｉ＝０は初期状態に対応する。
ｂ）　現在のパスのｉ番目の状態におけるｐの充足可能性を検査する。
ｃ）　充足可能な場合は、探索を成功として終了する。
ｄ）　充足不能な場合は、ｐはｉ番目の状態においては常に偽であることを学習し、この学習した知識を制約データベースに追加する。
ｅ）　ｉを所定の上限まで増大させながらステップｂ～ｄを繰り返し実行することによって探索を継続する。
ｆ）　所定の上限に到達すると、探索を不確定のまま終了する。
【請求項４１】
請求項４０に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、ステップｄ～ｅ間の完全性を検査するステップをさらに備える。
【請求項４２】
請求項４１に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって

、完全性を検査するステップはさらに下記ステップを備える。
（ｉ）　ｉ番目の状態を起点とする遷移がパス内の検査済みの状態を訪れないようにするための制約を追加し、制約データベースの充足可能性を検査する。
（ｉｉ）　ステップｉで充足不能な場合は、探索を失敗として終了する。
（ｉｉｉ）　ステップｉｉで充足不能な場合は、探索を継続する。および
（ｉｖ）　以前の状態がすべて検査されるまでステップ（ｉ）～（ｉｉｉ）を繰り返す。
【請求項４３】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、Ｘ（ｐ）の特性検査スキーマは下記ステップを備える。
ａ）　時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始し、ここでｉ＝０は初期状態に対応する。
ｂ）　さらにもう１つの時間フレームに遷移関係を展開し、時間フレームｉ＋１においてｐが真であるかどうかを検査する。
ｃ）　ｐが時間フレームｉ＋１において真であることが確認された場合は、成功として終了する。
ｄ）　ｐが時間フレームｉ＋１において偽であることが確認された場合は、失敗として終了する。
ｅ）　ｐの検査が時間フレームｉ＋１において不確定の場合は、不確定として終了する。
【請求項４４】
請求項４０に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、充足可能性検査のサブセットは単一の充足可能性検査として結合できる。
【請求項４５】
請求項４３に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、充足可能性検査のサブセットは単一の充足可能性検査として結合できる。
【請求項４６】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、真であることが既知の追加の制約を帰納不変式として使用する。
【請求項４７】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、帰納不変式として使用される過剰近似到達可能状態集合を得るために、近似到達可能性分析を使用する。
【請求項４８】
請求項４７に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、帰納不変式として使用される過剰近似到達可能状態集合を得るために、近似到達可能性分析をベースとする二分決定グラフを使用する。
【請求項４９】
請求項２８に記載の線形時間時相論理で表現された安全性特性の帰納証明の方法であって、展開されたモデル内で回路信号間の等価が発見され、帰納不変式として使用される。
【請求項５０】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、改変の下に埋め込まれた有界数のＥＸ演算子を有する特性を含めるために、充足可能性検査を使用してＥＸの解を列挙し、その後減少させることによって、特性検査スキーマが線形時間時相論理を超えて拡張される。
【請求項５１】
請求項５０に記載の任意線形時間論理時相特性の有界モデル検査方法であって、改変の下に埋め込まれたＦモダリティを近似的な方法で処理するために、有界数のＸ演算子を使用してＦをモデリングすることにより、特性検査スキーマが線形時間時相論理を超えて拡張される。
【請求項５２】
請求項１に記載の任意線形時間論理時相特性の有界モデル検査方法であって、特性二分決

定図ベース分析を使用して、有界モデル検査方式が計算木論理特性に拡張される。
【請求項５３】
任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、前記エンジンは下記要件を備える。
Ｆはｅｖｅｎｔｕａｌｉｔｙ演算子、Ｇはｇｌｏｂａｌｌｙ演算子、Ｕはｕｎｔｉｌ演算子、Ｘはｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）に関連付けられた特性を、ブーリアン充足可能性検査から成る特性検査スキーマに変換するように適応された特性トランスレーター、および、
特性全体を、Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）の各演算子に関する特性検査スキーマの反復呼び出しと、原子命題およびブーリアン演算子の標準的な処理で構成されるカスタマイズされた方法で検査するように適応された特性チェッカー。
【請求項５４】
請求項５３に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、次に検査する演算子に関して選択肢が存在する場合には、当該エンジンが、探索の難度により決定される優先順位に従って選択を行うように適応される。
【請求項５５】
請求項５４に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、前記探索の難度は、原子命題、Ｘ演算子、Ｆ演算子、Ｕ演算子、Ｇ演算子の順に高まる。
【請求項５６】
請求項５３に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、当該エンジンは、特性検査スキーマのサブセットに関して、対応するブーリアン充足可能性問題をサイズの小さい複数のブーリアン充足可能性下位問題に分割するために、対応する有界モデル検査問題のｋ番目のインスタンスに対して分割を実行するように適応される。
【請求項５７】
請求項５６に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、当該エンジンは、分割を演算子間にまたがって実行し、また各演算子に対しては時間フレーム間と時間フレーム内の両方で実行するように適応される。
【請求項５８】
請求項５７に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、当該エンジンは、分割を可能な場合は常に、漸増的に関連付けられるブール充足可能性下位問題に到達するように方向付けするように適応される。
【請求項５９】
請求項５７に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、当該エンジンは、複数のブール充足可能性下位問題にまたがって学習するように適応される。
【請求項６０】
請求項５７に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、当該エンジンは、複数の関連する下位問題にまたがる学習を行うために、ブール充足可能性アルゴリズムの漸増的な公式化を実行するように適応される。
【請求項６１】
請求項５３に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、当該エンジンは、定数の伝搬に基づく回路の単純化を実行するように適応される。
【請求項６２】
請求項５３に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証

用の検証エンジンであって、当該エンジンは、構造同形の検出に基づく回路の単純化を実行するように適応される。
【請求項６３】
請求項５３に記載の任意線形時間論理時相特性の有界モデル検査の機能を有する回路検証用の検証エンジンであって、当該エンジンは、ブール充足可能性検査問題を解決するためのハイブリッドＳＡＴソルバーをさらに備え、当該ハイブリッドＳＡＴソルバーは回路ベースおよびＣＮＦベースの充足可能性アルゴリズムの使用を組み合わせる。
【請求項６４】
線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、前記エンジンは下記要件を備える。
与えられた特性が否定された場合に、ｋインスタンスの基本ケースを検査するように適応された基本ケース・チェッカー、
与えられた特性に対応するモニター述語においてｋインスタンスの帰納ステップを検査するように適応された帰納ステップ・チェッカー、
否定された特性を特性に変換するように適応された特性トランスレーター、
Ｆはｅｖｅｎｔｕａｌｉｔｙ演算子、Ｘはｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）、Ｘ（ｐ）に関連付けられたブール充足可能性検査から成る検査スキーマ、および否定された特性の証拠を探索するように適応されたサーチャー。
【請求項６５】
請求項６４に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、次に検査する演算子に関して選択肢が存在する場合には、当該エンジンが、探索の難度により決定される優先順位に従って選択を行うように適応される。
【請求項６６】
請求項６５に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、難度が原子命題、Ｘ演算子、Ｆ演算子の順に高まると想定するように適応される。
【請求項６７】
請求項６４に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、特性検査スキーマのサブセットに関して、対応するブーリアン充足可能性問題をサイズの小さい複数のブーリアン充足可能性下位問題に分割するために、対応する有界モデル検査問題のｋ番目のインスタンスに対して分割を実行するように適応される。
【請求項６８】
請求項６７に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、分割を演算子間にまたがって実行し、また各演算子に対しては時間フレーム間と時間フレーム内の両方で実行するように適応される。
【請求項６９】
請求項６８に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、分割を可能な場合は常に、漸増的に関連付けられるブール充足可能性下位問題に到達するように方向付けするように適応される。
【請求項７０】
請求項６８に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、複数のブール充足可能性下位問題にまたがって学習するように適応される。
【請求項７１】
請求項６８に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、複数の関連する下位問題にまたがる学習を行うために、ブール充足可能性アルゴリズムの漸増的な公式化を実行するように適応さ

れる。
【請求項７２】
請求項６４に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、定数の伝搬に基づく回路の単純化を実行するように適応される。
【請求項７３】
請求項６４に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、構造同形の検出に基づく回路の単純化を実行するように適応される。
【請求項７４】
請求項６４に記載の線形時間論理時相特性のうち安全性特性を証明する機能を有する回路用帰納証明エンジンであって、当該エンジンは、ブール充足可能性検査問題を解決するためのハイブリッドＳＡＴソルバーをさらに備え、当該ハイブリッドＳＡＴソルバーは回路ベースおよびＣＮＦベースの充足可能性アルゴリズムの使用を組み合わせる。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明では、デジタル回路の形式的検証における技術に関し、特に、ブール充足可能性（Ｂｏｏｌｅａｎ　Ｓａｔｉｓｆｉａｂｉｌｉｔｙ）と二分決定グラフ（Ｂｉｎａｒｙ　Ｄｅｃｉｓｉｏｎ　Ｄｉａｇｒａｍｓ）に基づく新規な分割、順次学習、および帰納証明を使用して、デジタル回路の時相特性に関する有界モデル検査（ｂｏｕｎｄｅｄ　ｍｏｄｅｌ　ｃｈｅｃｋｉｎｇ）を行うためのソフトウェア、システム、方法が提供される。
【０００２】
【従来の技術】
１．　参考文献
以下の各論文は有益な背景情報を提供するものであり、それを理由として引用によりその全体がここに組み込まれ、本明細書の以降の部分では各々に付される括弧で囲んだ文献番号により適宜言及される。
【０００３】
【非特許文献１】
Ｅ．Ｍ．Ｃｌａｒｋｅ，０．Ｇｒｕｍｂｅｒｇ，ａｎｄ Ｄ．Ｐｅｌｅｄ，Ｍｏｄｅｌ Ｃｈｅｃｋｉｎｇ（モデル検査）：ＭＩＴ Ｐｒｅｓｓ，１９９９．
【非特許文献２】
Ｋ．Ｌ．ＭｃＭｉｌｌａｎ，Ｓｙｍｂｏｌｉｃ Ｍｏｄｅｌ Ｃｈｅｃｋｉｎｇ：Ａｎ Ａｐｐｒｏａｃｈ ｔｏ ｔｈｅ Ｓｔａｔｅ Ｅｘｐｌｏｓｉｏｎ Ｐｒｏｂｌｅｍ（状態爆発問題）：Ｋｌｕｗｅｒ Ａｃａｄｅｍｉｃ Ｐｕｂｌｉｓｈｅｒｓ，１９９３．
【非特許文献３】
Ａ．Ｂｉｅｒｅ，Ａ．Ｃｉｍａｔｔｉ，Ｅ．Ｍ．Ｃｌａｒｋｅ，ａｎｄ Ｙ．Ｚｈｕ，”Ｓｙｍｂｏｌｉｃ Ｍｏｄｅｌ Ｃｈｅｃｋｉｎｇ ｗｉｔｈｏｕｔ ＢＤＤｓ（ＢＤＤＳを使用しない記号的モデル検査），” ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ Ｗｏｒｋｓｈｏｐ ｏｎ Ｔｏｏｌｓ ａｎｄ Ａｌｇｏｒｉｔｈｍｓ ｆｏｒ Ａｎａｌｙｓｉｓ ａｎｄ Ｃｏｎｓｔｒｕｃｔｉｏｎ ｏｆ Ｓｙｓｔｅｍｓ（ＴＡＣＡＳ），ｖｏｌ．１５７９，ＬＮＣＳ，１９９９．
【非特許文献４】
Ｒ．Ｅ．Ｂｒｙａｎｔ，”Ｇｒａｐｈ－ｂａｓｅｄ ａｌｇｏｒｉｔｈｍｓ ｆｏｒ Ｂｏｏｌｅａｎ ｆｕｎｃｔｉｏｎ ｍａｎｉｐｕｌａｔｉｏｎ（ブール関数操作のためのグラフ・ベース・アルゴリズム），” ＩＥＥＥ Ｔｒａｎｓａｃｔｉｏｎｓ ｏｎ Ｃｏｍｐｕｔｅｒｓ，ｖｏｌ．Ｃ－３５（８），ｐｐ．６７７－６９１，

１９８６．
【非特許文献５】
Ｍ．Ｓｈｅｅｒａｎ，Ｓ．Ｓｉｎｇｈ，ａｎｄ Ｇ．Ｓｔａｌｍａｒｃｋ，”Ｃｈｅｃｋｉｎｇ Ｓａｆｅｔｙ Ｐｒｏｐｅｒｔｉｅｓ ｕｓｉｎｇ Ｉｎｄｕｃｔｉｏｎ ａｎｄ ａ ＳＡＴ Ｓｏｌｖｅｒ（帰納およびＳＡＴソルバーを使用した安全特性の検査），” ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｆｏｒｍａｌ Ｍｅｔｈｏｄｓ ｉｎ Ｃｏｍｐｕｔｅｒ－Ａｉｄｅｄ Ｄｅｓｉｇｎ，２０００．
【非特許文献６】
Ｐ．Ａｓｈａｒ，Ｍ．Ｇａｎａｉ，Ａ．Ｇｕｐｔａ，Ｚ．Ｙａｎｇ，Ａ．Ｍｕｋａｉｙａｍａ，ａｎｄ Ｋ．Ｗａｋａｂａｙａｓｈｉ，”Ｆｏｒｍａｌ Ｖｅｒｉｆｉｃａｔｉｏｎ ｉｎ ａｎ Ｉｎｄｕｓｔｒｉａｌ Ｓｅｔｔｉｎｇ：ＤｉＶｅｒ Ｖｅｒｉｆｉｃａｔｉｏｎ Ｐｌａｔｆｏｒｍ Ｗｈｉｔｅ Ｐａｐｅｒ（産業環境における形式的検証：ＤｉＶｅｒ検証プラットフォーム白書），” ＮＥＣ ＵＳＡ，ＣＣＲＬ ２００１－Ｃ００４－４－５５０９－１Ｎ，Ｄｅｃ ２００１ ２００１．
【非特許文献７】
Ｐ．Ａｓｈａｒ，Ｐ．Ｃｈａｕｈａｎ，Ｍ．Ｇａｎａｉ，Ａ．Ｇｕｐｔａ，Ｚ．Ｙａｎｇ，ａｎｄ Ｌ．Ｚｈａｎｇ，”ＤｉＶｅｒ Ｄｏｃｕｍｅｎｔ：Ｕｓｅｒ Ｍａｎｕａｌ Ｖｅｒｉｆｉｃａｔｉｏｎ Ｐｌａｔｆｏｒｍ ｆｏｒ Ｄｉｇｉｔａｌ Ｓｙｓｔｅｍｓ（ＤｉＶｅｒドキュメント：デジタル・システムのためのユーザー・マニュアル検証プラットフォーム），” ＮＥＣ ＵＳＡ，ＣＣＲＬ ２００１－Ｃ００５－４－５５０９－２Ｎ，Ｄｅｃ ２００１ ２００１．
【非特許文献８】
Ｐ．Ａｓｈａｒ，Ｐ．Ｃｈａｕｈａｎ，Ｍ．Ｇａｎａｉ，Ａ．Ｇｕｐｔａ，Ｚ．Ｙａｎｇ，ａｎｄ Ｌ．Ｚｈａｎｇ，”ＤｉＶｅｒ Ｄｏｃｕｍｅｎｔ：Ｔｕｔｏｒｉａｌ － ＮＥＣ ｐｒｏｄｕｃｔｉｏｎ ｄｅｓｉｇｎ ｃａｓｅ ｓｔｕｄｉｅｓ（ＤｉＶｅｒドキュメント：チュートリアル－ＮＥＣ生産設計事例研究），” ＮＥＣ ＵＳＡ，ＣＣＲＬ ２００１－Ｃ０８３－４－５５０９－５，Ｄｅｃ ２００１ ２００１．
【非特許文献９】
Ｈ．Ｚｈａｎｇ，”ＳＡＴＯ：Ａｎ ｅｆｆｉｃｉｅｎｔ ｐｒｏｐｏｓｉｔｉｏｎａｌ ｐｒｏｖｅｒ（ＳＡＴＯ：効率的な命題プルーバー），” ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ａｕｔｏｍａｔｅｄ Ｄｅｄｕｃｔｉｏｎ，ｖｏｌ．１２４９，ＬＮＡＩ，１９９７，ｐｐ．２７２－２７５．
【非特許文献１０】
Ｍ．Ｍｏｓｋｅｗｉｃｚ，Ｃ．Ｍａｄｉｇａｎ，Ｙ．Ｚｈａｏ，Ｌ．Ｚｈａｎｇ，ａｎｄ Ｓ．Ｍａｌｉｋ，”Ｃｈａｆｆ：Ｅｎｇｉｎｅｅｒｉｎｇ ａｎ Ｅｆｆｉｃｉｅｎｔ ＳＡＴ Ｓｏｌｖｅｒ（Ｃｈａｆｆ：効率的なＳＡＴソルバーの設計）ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ Ｄｅｓｉｇｎ Ａｕｔｏｍａｔｉｏｎ Ｃｏｎｆｅｒｅｎｃｅ，２００１．
【非特許文献１１】
Ｐ．Ｂｊｅｓｓｅ ａｎｄ Ｋ．Ｃｌａｅｓｓｅｎ，”ＳＡＴ－ｂａｓｅｄ ｖｅｒｉｆｉｃａｔｉｏｎ ｗｉｔｈｏｕｔ ｓｔａｔｅ ｓｐａｃｅ ｔｒａｖｅｒｓａｌ（状態空間トラバースを伴わないＳＡＴベース検証），” ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｆｏｒｍａｌ Ｍｅｔｈｏｄｓ ｉｎ Ｃｏｍｐｕｔｅｒ－Ａｉｄｅｄ Ｄｅｓｉｇｎ，２０００．
【非特許文献１２】
Ｍ．Ｇａｎａｉ ａｎｄ Ａ．Ａｚｉｚ，”Ｉｍｐｒｏｖｅｄ ＳＡＴ－ｂａｓｅ

d Bounded Reachability Analysis（改良型ＳＡＴベース有界到達可能性分析），” in Proceedings of VLSI Design Conference, ２００２.
【非特許文献１３】
O. Shtrichman, ”Tuning SAT Checkers for Bounded Model Checking（有界モデル検査用ＳＡＴチェッカーのチューニング），” in Proceedings of International Conference on Computer−Aided Verification, ２０００.
【非特許文献１４】
O. Shtrichman, ”Pruning Techniques for the SAT−based bounded model checking（ＳＡＴベース有界モデル検査のための枝取り），” in Proceedings of Workshop on Tools and Algorithms for the Analysis and Construction of Systems（ＴＡＣＡＳ），２００１.
【非特許文献１５】
M. Ganai, L. Zhang, P. Ashar, and A. Gupta, ”Combining Strengths of Circuit−based and CNF−based Algorithms for a High Performance SAT Solver（高性能ＳＡＴソルバーを得るための回路ベースおよびＣＮＦベース・アルゴリズムの強みの結合），” NEC USA, CCRL ２００１−Ｃ０７９−４−５５０２−２, December ２００１.
【非特許文献１６】
A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu, ”Symbolicmodel checking using SAT procedures instead of BDDs（ＢＤＤの代わりにＳＡＴプロシージャーを使用した記号的モデル検査），” in Proceedings of the Design Automation Conference, １９９９, pp. ３１７−３２０.
【非特許文献１７】
A. Gupta, Z. Yang, P. Ashar, and A. Gupta, ”ＳＡＴ−based Image Computation with Application in Reachability Analysis（到達可能性分析での応用例を持つＳＡＴベースのイメージ計算），” in Proceedings of Conference on Formal Methods in Computer−Aided Design, ２０００, pp. ３５４−３７１.
【非特許文献１８】
A. Gupta, A. E. Casavant, P. Ashar, X. G. Liu, A. Mukaiyama, and K. Wakabayashi, ”Property−specific testbench generation for guided simulation（誘導シミュレーションのための特性別テストベンチの生成），” in Proceedings of VLSI Design Conference, ２００２.
【非特許文献１９】
J. Kim, J. Whittemore, J. P. M. Silva, and K. Sakallah, ”IncrementalBoolean Satisfiability and its application to delay fault testing（増分ブール充足可能性とその遅延故障テストへの応用），” in Proceedings of International Works

hop on Logic Synthesis, 1999.
【非特許文献２０】
A. Kuehlmann and F. Krohm, ”Equivalence Checking using Cuts and Heaps（カットとヒープを使用した等価検査）,” in Proceedings of Design Automation Conference, 1997.
【非特許文献２１】
M. Ganai and A. Kuehlmann, ”On-the-fly compression of logical circuits（論理回路のオンザフライ圧縮）,” in Proceedings of International Workshop on Logic Synthesis, 2000.
【非特許文献２２】
A. Kuehlmann, M. Ganai, and V. Paruthi, ”Circuit-based Boolean Reasoning（回路ベースのブール推論）,” in Proceedings of Design Automation Conference, 2001.
【非特許文献２３】
P. Goel, ”An implicit enumeration algorithm to generate tests for Combinational circuits（組み合わせ回路用テストを生成するための暗黙エニュメレーション・アルゴリズム）” IEEE Transactions on Computers, vol. C-30, pp.
215-222, 1981.
【非特許文献２４】
H. Fujiwara and T. Shimono, ”On the Acceleration of Test Generation Algorithms（テスト生成アルゴリズムの加速）,” IEEE Transactions on Computers, vol. C-32, pp. 265-272, 1983.
【非特許文献２５】
M. Schulz, E. Trischler, and T. Sarfert, ”SOCRATES: A highly efficient ATPG System（SOCRATES: 高効率ＡＴＰＧシステム）,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 7, pp.
126-137, 1988.
【非特許文献２６】
M. N. Velev, ”Benchmark Suites.（ベンチマーク・スイート、http://www.ece.cmu.edu/~mvelev）,” October 2000.
【非特許文献２７】
A. Gupta, A. Gupta, Z. Yang, and P. Ashar, ”Dynamic detection and removal of inactive clauses in SAT with application in image computation（イメージ計算に応用事例を持つ、ＳＡＴにおける非活動文節の動的検出および除去）,” in Proceedings of Design Automation Conference, 2001.
【非特許文献２８】
H. Cho, G. D. Hachtel, E. Macii, B. Plessier, and F. Somenzi, ”Algorithms for approximate FSM traversal based on state s

pace decomposition, 状態空間分割に基づく近似ＦＳＭトラバーサルのためのアルゴリズム”ＩＥＥＥ Transactions on Computer－Aided Design of Integrated Circuits and Systems, vol. １５（１２）, pp. １４６５－１４７８, １９９６.
【非特許文献２９】
Ｉ.－Ｈ. Ｍｏｏｎ, Ｊ.－Ｙ. Ｊａｎｇ, Ｇ. Ｄ. Ｈａｃｈｔｅｌ, Ｆ. Ｓｏｍｅｎｚｉ, Ｃ. Ｐｉｘｌｅｙ, ａｎｄ Ｊ. Ｙｕａｎ, ”Approximate reachability don’t－cares for CTL model checking （ＣＴＬモデル検査のための近似到達可能性のドントケア）,” in Proceedings of the International Conference on Computer－Aided Design. San Jose, １９９８.
【非特許文献３０】
Ｋ. Ｒａｖｉ, Ｋ. Ｌ. ＭｃＭｉｌｌａｎ, Ｔ. Ｒ. Ｓｈｉｐｌｅ, ａｎｄ Ｆ. Ｓｏｍｅｎｚｉ, ”Approximation and decomposition of Decision Diagrams （決定図の近似と分解）,” in Proceedings of the Design Automation Conference. San Francisco, １９９８, pp. ４４５－４５０.

【０００４】
２． 序文
ａ） モデル検査
ハードウェア設計がますます複雑化するにつれ、コスト増大の原因となるエラーを回避できる効果的な検証方法へのニーズが高まっている。記号的モデル検査（非特許文献１、２参照）のような形式的検証手法の登場により、しらみ潰しの検査が可能になり、シミュレーションなどの従来手法に比較して、見つかりにくいバグも容易に検出できるようになった。しかし、これらの手法は、状態爆発問題が原因で実用使用においてはうまくスケーリングできないという難点があった。最近発表された代替手法では、有界モデル検査 （ＢＭＣ： Ｂｏｕｎｄｅｄ Ｍｏｄｅｌ Ｃｈｅｃｋｉｎｇ）（非特許文献３参照）が使用されている。主に二分決定グラフ（ＢＤＤ： Ｂｉｎａｒｙ Ｄｅｃｉｓｉｏｎ Ｄｉａｇｒａｍｓ）（非特許文献４参照）の使用をベースとする記号的モデル検査とは異なり、ＢＭＣはブール充足可能性 （ＳＡＴ： Ｓａｔｉｓｆｉａｂｉｌｉｔｙ） 決定手順の使用をベースとする。そのためＢＭＣでは、実用レベルではるかに大規模な設計を扱うことができ、特にバグが存在する場合に、その検出において高い効果を期待できる。
【０００５】
モデル検査では、ハードウェア設計は有限状態機械として表され、その仕様は時相論理で表現された特性で構成される。この仕様が充足されるかどうかを検査するため、設計の状態空間がしらみ潰しに探索される。仕様が充足されない場合は、典型的には、正当性特性の違反を実証する反証が検出される。有界モデル検査では、有界長ｋの反証（バグ）の検出に焦点が当てられる。これにより、無限パスと不動点演算の処理を回避できるほか、考慮の対象が有界長ｋのパスのみに絞り込まれるという利点が生まれる。具体的には、この問題は、長さｋの反証が存在している場合にのみ真となる命題式に変換される。実用使用では、境界ｋを漸増させて最も短い反証を検出することができる。ただし、一定の境界まで反証が検出されないときの完全性を保証するため、これとは別に推論を行う必要がある（非特許文献３、５参照）。
【０００６】
【発明が解決しようとする課題】
３． 技術と関連研究の背景
この項では、関連研究の様々な態様を、背景とＢＭＣの従来技術を踏まえて論じる。

【０００７】
仕様はＬＴＬ（Ｌｉｎｅａｒ Ｔｅｍｐｏｒａｌ Ｌｏｇｉｃ：線形時相論理）で表現される。ＬＴＬには、ｎｅｘｔ ｔｉｍｅ演算子Ｘ、ｅｖｅｎｔｕａｌｉｔｙ演算子Ｆ、ｇｌｏｂａｌｌｙ演算子Ｇ、ｕｎｔｉｌ演算子Ｕの各時相演算子が含まれる。解説を簡素化するため、ここではタイプＥ ｆの式について考察する。ここで、Ｅは存在パス修飾子、ｆはパス修飾子を持たない時間式である。設計は、状態の有限集合Ｓ、初期状態の集合Ｉ、状態Ｔ間の遷移関係、原子命題を有する状態のラベリングＬで構成される、標準Ｋｒｉｐｋｅ構造Ｍ ＝（Ｓ，Ｉ，Ｔ，Ｌ）として記述される。Ｋｒｉｐｋｅ構造Ｍ、ＬＴＬ式ｆ、境界ｋが与えられたとすると、ＢＭＣにおける変換タスクとは、すなわち、構造Ｍ内の式ｆに関して長さｋの証拠が存在するときにのみ充足可能な命題式＜Ｍ，ｆ＞$_k$を構築することとなる。

【０００８】
この状態集合は有限なので、状態はブール変数ｓの記号エンコードを使用して表され、長さｋの有限シーケンスはシーケンスｓ$_0$ ．．． ｓ$_k$として表される。

【０００９】
さらに、このシーケンスがＭ内で有効なパスであることを保証するために、第１の制約集合＜Ｍ＞$_k$が使用される。

【数１】

$$[M]_k = I(s_0) \wedge \wedge_{i=0}^{i=k-1} T(s_i, s_{i+1})$$

【００１０】
この式の最初の部分は「シーケンスにおける第１の状態は初期状態Ｉでなくてはならない」という制約を課し、この式の２番目の部分は「連続する状態の各ペアは遷移関係Ｔに準じた関係になければならない」という制約を課す。ここでは、この２番目の部分は順次設計のｋステップにわたる展開に対応し、ｋの増大に伴ってＳＡＴ問題のサイズが増大することに注意する必要がある。

【００１１】
その次の制約集合では、Ｍ内のこの有効パスがＬＴＬ式ｆを充足することが保証される。また、図２（非特許文献３より）に示されるように、このパスが（ｋ，ｌ）ループかどうかによって、ケース分割が必要となることもある。状態ｋから状態ｌへのループのケースは、$_l$Ｌ$_k$ ＝Ｔ（ｓ$_k$，ｓ$_l$）に変換できる。ループがないケースは、¬Ｌ$_k$に変換できる。ここで、

【数２】

$$L_k = \vee_{l=0}^{l=k} {}_lL_k$$

である。［ｆ］^{0} $_k$はループなしケースにおける式ｆの変換を表し、$_l$［ｆ］^{0} $_k$は（ｋ，ｌ）ループのケース（詳細な定義については非特許文献３を参照）におけるｆの変換を表す。最後に、問題全体の一括変換は以下によって得られる。

【数３】

$$[M,f]_k = [M]_k \wedge ((\neg L_k \wedge [f]_k^0) \vee (\vee_{l=0}^{l=k} ({}_lL_k \wedge {}_l[f]_k^0)))$$

【００１２】
ここで、１番目の項は有効パスの要件を示し、２番目の項は、そのパスがケース分割された仕様式を充足することを示す。本論文の解説においては、この変換がモノリシックであ

る（すなわち、与えられたｋに対して式全体が生成される）ことに留意することが重要である。この式は、ＳＡＴＯ（非特許文献９参照）、Ｃｈａｆｆ（非特許文献１０参照）などの標準的なＳＡＴソルバーによる充足可能性検査に付される。

【００１３】

実用使用では、長い証拠の探索は境界ｋを漸増させて実行される。これは、証拠が存在する場合にはきわめて有効である。しかし、証拠が存在しない場合には、その特性が偽であることを確定するために別の証明手法が必要となる。この場合は、有限状態機械の直径までのすべてのｋを検査するので十分である（非特許文献３参照）。また、同様の状況において安全特性を証明するための方法として、ループなしパス、最短パスなどその他の制約の使用も提唱されている（非特許文献５、１１参照）。

【００１４】

ＢＭＣエンジンは、有界長の反証を検出することに加えて、帰納証明の実行にも使用できる（非特許文献１６参照）。深度ｋを増大させて実行される帰納と、ループなしパスに対する制限を組み合わせると、安全特性の完全な証明手法が得られる（非特許文献５、１１参照）。深度ｋの帰納は、以下の２ステップから成る。

【００１５】

（ａ）　基本ケース：その特性が、初期状態から始まる長さｋの各パス上で有効であることを証明する。

【００１６】

（ｂ）　帰納ステップ：その特性が、任意の状態から始まる長さｋのパス上で有効であり、さらには、長さ（ｋ＋１）のパスの拡張上でも有効であることを証明する。

【００１７】

ループなしパスに限定することにより、「パス内に同一の状態がない」という追加の制約が課される。ここで、基本ケースでは初期状態制約が使用されるが、帰納ステップではこの制約は使用されないことに注意する必要がある。したがって、帰納ステップには到達不能な状態も含まれる可能性がある。実用使用では、追加の制約　（すなわち、特性自体よりも強力な帰納不変式）を使わずに帰納証明を完了することは不可能なこともある。この場合は、到達可能性の制約など、設計者に既知の回路制約を使用して帰納不変式を強化することができる。

【００１８】

上記の手法は、ゲートとフリップフロップが多数存在する回路上では演算能力が低下する。これらの手法は回路のサイズが大きくなるに従って急激に低速となり、メモリ消費量も増大する。

【００１９】

本発明は、従来技術に関連する短所を克服し、上記問題のいくつかを解決することを目指すものであり、線形時間論理で表された任意の時相特性を対象とした有界モデル検査の方法を提供することを目的とする。

【００２０】

【課題を解決するための手段】

本発明による有界モデル検査の方法は、Ｆはｅｖｅｎｔｕａｌｉｔｙ演算子、Ｇはｇｌｏｂａｌｌｙ演算子、Ｕはｕｎｔｉｌ演算子、Ｘはｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）に関連付けられた特性を、ブール充足可能性検査から成る特性検査スキーマに変換するステップから成る。特性全体は、Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）の各演算子に関する特性検査スキーマの反復呼び出しと、原子命題およびブール演算子の標準的な処理で構成されるカスタマイズされた方法で検査される。

【００２１】

ある特定の機能拡張では、次に検査する演算子に関して選択肢が存在する場合には、探索の難度により決定される優先順位に従って選択が行われる。

【００２２】

さらに具体的には、この探索の難度は、原子命題、Ｘ演算子、Ｆ演算子、Ｕ演算子、Ｇ演算子の順に高まると想定される。

【００２３】

他の特定の機能拡張では、特性検査スキーマのサブセットが、対応するブール充足可能性問題をサイズの小さい複数のブール充足可能性下位問題に分割するために、対応する有界モデル検査問題のｋ番目のインスタンスに対して分割を実行する。

【００２４】

さらに具体的には、この分割は演算子間にまたがって実行され、各演算子に対しては時間フレーム間と時間フレーム内の両方で実行される。

【００２５】

さらに具体的には、この分割は可能な場合は常に、漸増的に関連付けられるブール充足可能性下位問題に到達するように方向付けされる。

【００２６】

さらに具体的には、複数のブール充足可能性下位問題にまたがる学習が使用される。

【００２７】

さらに具体的には、複数の関連する下位問題にまたがる学習を行うために、ブール充足可能性アルゴリズムの漸増的な公式化が使用される。

【００２８】

他の特定の機能拡張では、定数の伝搬に基づく回路の単純化が使用される。

【００２９】

他の特定の機能拡張では、構造同形の検出に基づく回路の単純化が使用される。

【００３０】

他の特定の機能拡張では、ブール充足可能性検査問題が、回路ベースとＣＮＦベースの充足可能性アルゴリズムの使用を結合したハイブリッドＳＡＴソルバーを使用して解決される。

【００３１】

他の特定の機能拡張では、Ｆ（ｐ）の特性検査スキーマは、時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始するステップを備え、ここでｉ＝０は初期状態に対応する。現在のパスのｉ番目の状態におけるｐの充足可能性が検査される。充足可能な場合、探索は成功として終了される。充足不能な場合、ｐはｉ番目の状態においては常に偽であることが学習され、この学習された知識が制約データベースに追加される。探索は、ｉを所定の上限まで増大させながらそれまでのステップを繰り返し実行することで継続される。所定の上限に到達すると、探索は不確定のまま終了される。

【００３２】

他の特定の機能拡張では、完全性が検査される。

【００３３】

さらに具体的には、この完全性の検査は、ｉ番目の状態を起点とする遷移がパス内の検査済みの状態を訪れないようにするための制約を追加するステップと、制約データベースの充足可能性を検査するステップとを備える。充足不能な場合、探索は失敗として終了される。充足不能でない場合、探索は継続される。これらのステップは、以前の状態がすべて検査されるまで繰り返される。

【００３４】

他の特定の機能拡張では、Ｇ（ｐ）の特性検査スキーマは、時間フレームｉにおいて所定の制約データベースを使用して所定の開始状態から探索を開始するステップを備え、ここでｉ＝０は初期状態に対応する。ｐが現在のパスのｉ番目の状態において充足されることを保証するための制約がデータベースに追加され、充足可能性が検査される。充足不能な場合、探索は失敗として終了される。充足可能な場合、開始状態からｉ番目の状態までの間でｊ番目の状態が各々ループバック状態かどうか検査され、ループバック状態が検出されたら探索は成功として終了される。開始状態より前のｊ番目の状態が各々ループバック

状態かどうか検査され、ループバック状態が検出されたら探索は成功として終了される。探索は、ｉを所定の上限まで増大させながら、これらのステップを繰り返し実行することで継続される。所定の上限に到達した時点で、探索は不確定のまま終了される。

【００３５】

前述の方法の機能拡張には上記に加えてさらに数種類があることは、請求項と発明の詳細な説明から明らかとなるであろう。

【００３６】

本明細書における発明の他の態様は、線形時間時相論理で表現された安全性特性の帰納証明の方法であって、ｋインスタンスの基本ケースを、与えられた特性の否定に関して検査するステップ、ｋインスタンスの帰納ステップを、与えられた特性に対応するモニター述語に関して検査するステップ、ｋ＝０から開始し、任意に指定された上限までそれを漸増させるステップを備え、基本ケースの検査はさらに、否定された特性を、Ｆがｅｖｅｎｔｕａｌｉｔｙ演算子、Ｘがｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）およびＸ（ｐ）と関連づけられたブール充足可能性検査から成る特性検査スキーマに変換するステップを備える。否定された特性の証拠を検出するために、Ｆ（ｐ）、Ｘ（ｐ）に関する特性検査スキーマの反復呼び出しと、原子命題およびブール演算子の標準的な処理で構成される探索が初期状態を起点として実行される。証拠が検出されないことが証明された場合は、証明全体が成功として終了される。ｋ時間フレーム内に証拠が検出された場合は、証明全体が失敗として終了され、結果が不確定の場合は、以下の特性検査スキーマから成る帰納ステップの検査が継続される。すなわち、モニター述語が任意状態からｋ時間フレームにわたって有効で、かつｋ＋１番目の時間フレーム内では有効ではない状況を保証するための制約が追加される。充足不能な場合、探索は成功として終了される。充足不能でない場合、探索は継続される。この探索は、ｋを所定の上限まで漸増させながら基本ケースと帰納ステップを繰り返し実行することで継続される。ｋが指定された上限に到達した場合は、証明は不確定となる。

【００３７】

本明細書における発明の他の態様は、回路を検証するための検証エンジンであって、このエンジンは線形時間論理で表された任意の時相特性の有界モデル検査を実行する機能を有する。このエンジンは、特性トランスレーターと特性チェッカーを備える。特性トランスレーターは、Ｆはｅｖｅｎｔｕａｌｉｔｙ演算子、Ｇはｇｌｏｂａｌｌｙ演算子、Ｕはｕｎｔｉｌ演算子、Ｘはｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）に関連付けられた特性を、ブール充足可能性検査から成る特性検査スキーマに変換するように適応される。特性チェッカーは、特性全体が、Ｆ（ｐ）、Ｇ（ｐ）、Ｕ（ｐ，　ｑ）、Ｘ（ｐ）の各演算子に関する特性検査スキーマの反復呼び出しと、原子命題およびブール演算子の標準的な処理で構成されるカスタマイズされた方法で検査されるように適応される。

【００３８】

本明細書における発明の他の態様は、回路のための帰納証明エンジンであって、このエンジンは線形時間論理の時相特性における安全性特性を証明する機能を有する。このエンジンは、与えられた特性が否定された場合に、ｋインスタンスの基本ケースを検査するように適応された基本ケース・チェッカーを備える。与えられた特性に対応するモニター述語においてｋインスタンスの帰納ステップを検査するように適応された、帰納ステップ・チェッカーが提供される。否定された特性を、Ｆがｅｖｅｎｔｕａｌｉｔｙ演算子、Ｘがｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）およびＸ（ｐ）と関連づけられたブール充足可能性検査から成る特性検査スキーマに変換するように適応された特性トランスレーターが提供され、さらに、否定された特性の証拠を探索するように適応されたサーチャーが提供される。

【００３９】

本明細書における発明のさらに他の態様は、回路のための帰納証明エンジンであって、このエンジンは線形時間論理の時相特性における安全性特性を証明する機能を有する。この

エンジンは、与えられた特性が否定された場合に、ｋインスタンスの基本ケースを検査するように適応された基本ケース・チェッカーを備える。与えられた特性に対応するモニター述語においてｋインスタンスの帰納ステップを検査するように適応された、帰納ステップ・チェッカーが提供される。否定された特性を、Ｆがｅｖｅｎｔｕａｌｉｔｙ演算子、Ｘがｎｅｘｔ－ｔｉｍｅ演算子を表す時相演算子Ｆ（ｐ）およびＸ（ｐ）と関連づけられたブール充足可能性検査から成る特性検査スキーマに変換するように適応された、特性トランスレーターが提供される。否定された特性の証拠を探索するように適応された、サーチャーが提供される。

【００４０】

上記の様々な特定の機能拡張に関してさらに具体的な機能拡張が提供されることは、請求項と詳細な説明から明らかとなるであろう。

【００４１】

【発明の実施の形態】

（カスタマイズ特性変換）

この節では、本発明の一部を成すカスタマイズ変換について説明する。この変換スキーマは、モノリシックＳＡＴ式を生成するのではなく、可能な場合は早期に終了することを前提に、有界な論理積／論理和に対して遅延指標付けを行うことにより、式を漸増的に構築するものと考えることができる。学習と分割を使用して複数の単純なＳＡＴ下位問題を生成するステップを採用した、漸増的な公式化は、広く使用されている。

【００４２】

場合によっては、ループなしパスのスケルトン（パス上のすべての状態が特異な対を成す）にのみ焦点を当てることを目的として、追加の制約が使用されることもある。これらの制約は、安全特性を対象とする場合には、証明の完全性を保証するために使用されていた（非特許文献５、１１参照）。変換スキーマでは、これらの制約は、ループを有する証拠を必要とする活性特性を対象に使用される。一例として、特性Ｇ（ｐ）について考察する。各状態においてｐが真であり、パス上の最後の状態を除くすべての状態が特異な場合には、ループなしパスのスケルトンに焦点が当てられる。最後の状態は、「ループバック状態」と呼ばれるパス上のｌ番目の状態と同一であり、これによって（ｋ，ｌ）ループを形成する。

【００４３】

Ｇ（ｐ）の証拠はすべて、境界ｋを増大させることによってシステマティックに検出される、ループなしスケルトンを持つことは明白である。ｋが与えられた場合、ループバック状態の探索は初期状態かｋ番目の状態のどちらからでも開始できる。ループバック状態が検出された場合、特性は真である。一方、すべてのループなしスケルトンを検査してもループバック状態が検出されなかった場合には、特性は偽である。Ｇ（ｐ）特性の一括変換もまた、ｋを増大させて（ｋ，ｌ）ループを考慮することはすでに述べた。ただし、この変換では、「パス・スケルトンがループなしでなければならない」という制約は課せられない。ループなしパスのスケルトンには、設計の順次直径ほどに厳格な境界はないが、ＳＡＴソルバーの範囲内での証明機能が提供される。これに対し、順次直径に関する推論には、ＱＢＦ（Ｑｕａｎｔｉｆｉｅｄ　Ｂｏｏｌｅａｎ　Ｌｏｇｉｃ：　定量化ブール論理）ソルバーが必要とされる。

【００４４】

詳細を説明する前に、本発明の変換スキーマにより生成されるＳＡＴ下位問題を定義する数種類の制約を分類しておくと有益である。これらの制約とは、以下のとおりである。

－　　回路制約
－　　特性副次式による制約
－　　充足不能なＳＡＴインスタンスから学習された制約
－　　ループなしパスのスケルトンのみを考慮するためのループ検査制約

【００４５】

一括変換では、最初の２種類の制約　（すなわち、回路制約と特性制約）しか使用されな

いことに注意する必要がある。学習された制約（非特許文献１２参照）とループ検査制約（非特許文献５参照）の使用は、各々従来から知られていたが、単純な安全特性を検証する状況での使用に限られている。また、典型的にはＳＡＴソルバーそのものによって生成される、他の１種類の制約（矛盾する文節から生じる制約）があることにも注意する必要がある。本発明の変換スキームでは、漸増的な公式化を使用することにより、こうした制約も容易に共有することができる。

【００４６】

本発明のカスタマイズ特性変換における分割、学習、およびインクリメンタル（漸増的）の各態様への注目を促すため、一般的なＬＴＬ特性を処理するための擬似コードの非限定的な例をここに示す。ここで、ｐおよびｑは、命題式とＸ演算子との任意のブール結合を示し、各々は回路のグラフ表現における１つのノードと関連付けられる。ｉｓ＿ｓａｔ（Ｃ）は、ＳＡＴソルバーへの呼び出しを示し、ＳＡＴソルバーは、ブール式Ｃが充足可能である場合にのみ真を返す。Ｌ＿ｉｊは、ｉ番目からｊ番目までの時間フレームの間にループ遷移がある（すなわち、Ｌ＿ｉｊ＝Ｔ（s_i，s_j））ことを示す。Ｎは、ユーザーの制御下にある展開の最大深度を示す。説明を簡素化するため、これらの変換では、ＳＡＴ下位問題に常に追加される回路制約は示していない。

【００４７】

一例として、Ｆ（ｐ）の変換について考慮する。指標ｉにおける外側ｆｏｒループ（第１３行）は、ＢＭＣの境界ｋをユーザー指定の上限Ｎまで漸増することに相当する。このループは、特性データベースＣを構築する（第８行）。このデータベースは、初期には真である。各ｉについて、（Ｃ　＆　ｐ＿ｉ）の充足可能性（すなわち、現在の時間フレームにおいてｐが真である間に、現在の文節データベースが充足可能かどうか）が検査される（第１４行）。充足可能な場合、Ｆ（ｐ）の証拠が検出され、真が返される。充足可能でない場合は、「時間フレームｉにおいてｐは常に偽である」という事実が学習され、文節データベースＣ（第１５行）に追加される。指標ｊにおける内側ｆｏｒループ（第１６～２１行）は任意であり、証明を実行する際に選択的に使用される。このループは、パス上の現在の状態s_iから任意の以前の状態s_jまでの間にループがないことを保証するために、対を成す制約をＣに追加する。続いて、Ｃ上でＳＡＴ検査が実行される（第２１行）。Ｃが充足不能な場合は、どの時間フレームにおいても、すべてのループなしパスを検査した結果ｐが真であることが検出されなかったことを意味するので、その特性は偽であると結論付けることができる。これにより、Ｆ（ｐ）に関する完全性の引数が得られる。一方、Ｃが充足可能な場合は、現在のパスをループなしに保ちながら拡張する方法があるので、ループ指標ｉを漸増させながらさらに繰り返しが実行される。さらなる最適化の方策として、第１９行のコメント行に示されるように、対を成すループ制約が漸増的に追加される。これにより、すべての制約を追加することなく早期に終了できる可能性が生まれる。

【００４８】

ループ付きの証拠を必要とする特性Ｇ（ｐ）に関して、ループ制約がどのように処理されるかを知ることも有益である。ここでも、ｉにおける外側ｆｏｒループは、ＢＭＣの境界ｋを漸増させることに相当する。このプロシージャーは、（Ｃ　＆　ｐ＿ｉ）の充足可能性を検査することから開始される。充足不能な場合は、現在の時間フレームｉにおいてｐを充足する方法はないので、特性Ｇ（ｐ）は偽である。充足可能な場合は、このパス上の現在の状態s_iから以前の状態s_jまでの間にループ遷移があるかどうか検査するための制約が追加される。これは漸増的に検査されるので（第３５行）、該当する最初のループが検出された時点でプロシージャーを終了できる。該当するループが検出されない場合は、この事実が学習され、文節データベースＣに追加される（第３６行）。すべてのループ制約が追加されたら、完全性検査が実行され（第３８行）、これによって、特性が偽であると結論付けるか、あるいは次の繰り返しのためにループなしパスのスケルトンを拡張することが可能になる。以下に、一般的なＬＴＬ特性に関する疑似コードを示す。

【００４９】

```
1  /* N = Max depth
2    f_i = property node at i^th time-frame
3    L_ij = Loop constraint between the i^
h 4                and j^th time frames
5    &(l) = cnf conjunction(disjunction) *
/6
7    F(p) {
8      F(1, p, 0)
9    }
10                                                    10
11   F(IC, p, start) {
12     C = IC ;
13     for( i = start ; i < N ; i++) {
14       if( is_sat(C & p_i)) return true ;
15       C = C & !p_i ;
16       for( j = i ; j >= start ; j--) { //
optional 16-2117          C = C & !L_ij ;
18         // optional 19
19         // if( !is_sat(C)) return false;2
0       }                                             20
21       if ( !is_sat(C)) return false;22
 }
23   }
24
25   G(p) {
30       C = 1;
31       for ( i = 0 ; i < N ; i++) {
32         C = C & p_i ;
33         if ( !is_sat(C)) return false ;34
       for ( j = i ; j >= 0 ; j-- ) {             30
35         if ( is_sat( C & L_ij )) return t
rue ;36          C = C & !L_ij ;
37       }
38       if ( !is_sat(C)) return false ;39
   }
40   }
41
42   G( IC, p, start) {
43       C = IC ; t = 0 ; C'' = 1 ;
44       for ( i = start ; i < N ; i++) {45    40
       C = C & p_i ;
46         if ( !is_sat(C)) return false ;47
       for ( j = i ; j >= start ; j-- ) {4
8          if ( is_sat( C & L_ij )) return tr
ue ;49          C = C & !L_ij ;
50       }
51       C' = C & C'' ;
52       for ( j = start -1 ; j >= t ; j--)
53         if ( !is_sat( C' & p_j )) {
54           t = j + 1 ; C'' = C'' & !p_j ; b   50
```

```
reak ; } 55         C' = C' & p_j ;
56          if ( is_sat( C' & L_ij ) return t
rue57         C' = C' & !L_ij ;
58        }
59        for ( l = start -1 ; l >= t ; l--)
60          C = C & !L_il ;
61          if ( !is_sat(C)) return false ;62
      }
63    }
64
65    F( p or F(q) ) {
66        C = 1 ;
67        for ( i = 0 ; i < N ; i++) {
68          if ( is_sat( C & p_i ) return tru
e ;69           C = C & !p_i ;
70
71          if ( F(C, q, i)) return true ;
72
73          for ( j = i ; j >= 0 ; j--) {74
      C = C & !L_ij ;
75          // if ( !is_sat(C)) return false
;76          }
77        // reached the bound
78        if ( !is_sat(C)) return false ;79
    }
80    }
81
82    F( p or F(q) ) {
83        C = 1 ;
84        for ( i = 0 ; i < N ; i++) {
85          if ( !is_sat( C & p_i ) {
86            C = C & !p_i ;
87          } else {
88          if ( F(C & q, i, q, i )) return true
 ;89          }
90          for ( j = i ; j >= 0 ; j--) {91
      C = C & !L_ij ;
92          // if ( !is_sat(C)) return false
;
93          }
94        // reached the bound
95        if ( !is_sat(C)) return false ;96
   }
97    }
98
99    F( p or G(q) ) {
100       C = 1 ;
101       for ( i = 0 ; i < N ; i++) {
102         if ( is_sat( C & p_i ) return fal
se ;103           C = C & !p_i ;
```

```
104
105       if ( G(C, q, i )) return true ;106
107       for ( j = i ; j >= 0 ; j--) {108
      C = C & !L_ij ;
109       // if ( !is_sat(C)) return false
;110       }
111     // reached the bound
112     if ( !is_sat(C)) return false ;113
   }
114  }
115
116    F( p or G(q) ) {
117       C = 1 ;
118       for ( i = 0 ; i < N ; i++) {
119         if ( !is_sat( C & p_i ) {
120           C = C & !p_i ;
121         } else {
122         if ( G(C & p_i, q, i )) return tru
e ;123         }
124         for ( j = i ; j >= 0 ; j--) {125
        C = C & !L_ij ;
126         // if ( !is_sat(C)) return false
  ;127         }
128       // reached the bound
129       if ( !is_sat(C)) return false ;13
0     }
131   }
132
133    U( IC, q, r, start ) {
134       C = IC ;
135       for ( i = start ; i < N ; i++) {13
6       if ( is_sat( C & r_i )) return tru
e ;137         C = C & !r_i & q ;
138         if ( !is_sat(C)) return false ;1
39         for ( j = i ; j >= 0 ; j--) {140
       C = C & !L_ij ;
141         // if ( !is_sat(C)) return false
  ;142         }
143       // reached the bound
144       if ( !is_sat(C)) return false ;14
5     }
146   }
147
148    F( p or U(q, r) ) {
149     C = 1 ;
150     for ( i = 0 ; i < N ; i++) {
151       if ( !is_sat( C & p_i ) {
152       C = C & !p_i ;
153      } else {
154         if ( U(C & p_i, q, r, i )) return t
```

```
rue ;155       }
156       for ( j = i ; j >= 0 ; j--) {157
      C = C & !L_ij ;
158       // if ( !is_sat(C)) return false
  ;159       }
160       // reached the bound
161       if ( !is_sat(C)) return false ;1
62       }
163    }
165    F( p or U(q, r) ) {
166      C = 1 ;
167      for ( i = 0 ; i < N ; i++) {
168        if ( is_sat( C & p_i )) return t
rue ;169        C = C & !p_i ;
170        if ( U(C & q, r, i )) return true
;171        for ( j = i ; j >= 0 ; j--) {172
      C = C & !L_ij ;
173        // if ( !is_sat(C)) return false
  ;174       }
175      // reached the bound
176       if ( !is_sat(C)) return false ;17
7     }
178    }
```

【００５０】

Ｆ（ｐ ａｎｄ Ｇ（ｑ））のように特性をネストすると、構成要素のＦ副次式とＧ副次式に分割することができ、そのパーティション全体にまたがる学習が可能になる。ここではＦ（ｐ ａｎｄ Ｇ（ｑ））を対象に、非限定アルゴリズムの例を示す。ｉにおける外側ループ（第１１８行）は、ＢＭＣの境界ｋを漸増させる。第１のＳＡＴ下位問題は、（Ｃ ＆ ｐ＿ｉ）を検査する。偽の場合は、ｔｈｅｎ ！ｐ＿ｉが学習され、Ｃに追加される（第１２０行）。このとき、現在のパスをループなしのまま拡張できなければ（第１２４～１２８行）、特性が偽であることが証明される。一方、（Ｃ ＆ ｐ＿ｉ）が真であれば、アルゴリズムは時間フレームｉからＧ（ｑ）の検査を継続する。

【００５１】

この検査は、関数呼び出しＧ（Ｃ ＆ ｐ＿ｉ，ｑ，ｉ）によって実行される。Ｇ（Ｃ ＆ ｐ＿ｉ、ｑ、ｉ）は、s_iを起点としてパスの探索を行い、各状態でｑが真であることが確認されると、パスが以前の状態にループバックする。ここで、Ｇ（Ｃ ＆ ｐ＿ｉ，ｑ，ｉ）は、時間フレームｉよりも短い時間フレームｊにおいてもループバック状態を検査する（第５２～５８行）ことに注意する必要がある。この場合、Ｇ（Ｃ ＆ ｐ＿ｉ，ｑ，ｉ）はさらにs_iまで、各状態においてｑが真であることを追加して証明しなければならない。任意の時間フレームｊにおいて充足不能なことが判明した場合は、！ｑ＿ｊが学習される（第５４行）。ここで、この特性を分析する過程で多数のＳＡＴ呼び出しが行われるという事実と、データベース（Ｃ，Ｃ’，Ｃ”）が連続する呼び出しの間でどのように漸増されるかについて注目する必要がある。ＳＡＴ呼び出しで充足不能であることが判明するごとに、新たな制約を学習する機会が得られる。こうした特性の分析は、一括変換を使用するよりも、本発明の手法を使用した方がはるかに高速になる。

【００５２】

擬似コードに示される残りの特性と、有益とみなされるその他のＬＴＬ特性は、同様の方法で処理される。この手法の特徴は、可能な場合は常に、分割、学習、および漸増的ＳＡＴ検査が実行されることにある。また、証明の完全性を確保するため、ループ検査の制約

を任意に追加することもできる。

【００５３】

１．　正当性特性の拡張範囲

候補となるすべての正当性特性がＬＴＬで表現できるわけではない。例えば、生産設計の１つにおいて、仕様ＡＧ（ｐ　－＞ＥＸ　ｑ）を使用して、任意の状態を起点とする１つの遷移の実現可能性を検査する必要が生じたことが報告されている。これはＬＴＬ特性として表現することはできないが、ＳＡＴベースのＢＭＣフレームワーク内でそれを処理することは可能である。

【００５４】

以下に、有界数のＥＸ演算子（「ＥＸ：ｎ」として示す）を使用した、さらに一般的なケースを疑似コードで示す。非限定的な例として、第１の変換について考慮する（第２の変換は、第１の変換と同様である）。第１の変換は、否定された特性ＥＦ（ｐ　＆　！（ＥＸ　ｑ））の証拠を探索するので、前述の特性の反証を得るために使用できる。ここで注目すべきは、！（ＥＸ　ｑ）がＥＸ　！ｑと同一となる可能性のあるＬＴＬセマンティクスはここでは使用されないことである。その代わりに、！（ＥＸ　ｑ）＝　ＡＸ　！ｑであるＣＴＬセマンティクスが使用される。順次設計をさらにｎステップにわたって前方に展開し、Ｘ：ｎ　ｑに対応する回路ノードを作成することにより（「ｑ＿｛ｎ＋１｝」として示される）、ｎ個のＸ演算子を処理することは容易である。以下に、一般的な非ＬＴＬ特性に関する疑似コードを示す。

【００５５】

```
1  EF(p and ! EX:n q) {
2    C = 1; C' = 0;
3    for (i=0; i<N; i++) {
4      if (!is_sat(C & p_i)) {
5        C = C & !p_i;
6      } else {
7        if (!is_sat(C & p_i & q_{n+i})) 8
     return true;
9        else {
10         // exclude states with p & q_{n+i
} 11         while(1) {
12           C' = C'|get_sat_state_cube(i);13
           if (!is_sat(C & !C' & p_i))
14             break; // no more p_i states15
           if (!is_sat(C & !C' & p_i & q_{n+i
}))16               return true; // found witne
ss17         } // end of while loop
18       }
19     }
20     for (j=i; j>=0; j--) {
21       C=C & !L_ij;
22       // if (!is_sat(C)) return false;2
3    }
24     if (!is_sat(C)) return false;
25   }
26 }
27
28 ef(p or !EX:n q) {
29   C=1;
30   for (i=0; i<N; i++) {
```

```
31    if (is_sat(C & p_i)) return true;32
    else {
33      C=C & !p_i;
34      if (!is_sat(C & q_{n+i})) return
else;35        else {
36        // excluse states with q_{n+i}37
      while(1) {
38        C'=C'|get_sat_state_cube(i);39
      if (!is_sat(C & !C')
40          break; // no more states
41        if (!is_sat(C & !C' & q_{n+i}))4
2          return true; // found witness
43      } // end of while loop
44    }
45   }
46    for (j=i;j>=0;j--) {
47      C=C & !L_ij;
48      // if (!is_sat(C)) return false;4
9    }
50    if (!is_sat(C)) return false;
51  }
52 }
```

【００５６】

このプロシージャーは、ｐ＿ｉが充足可能かどうか検査する（第４行）。充足可能でない場合は、！ｐ＿ｉ（第５行）が学習され、ｉが漸増される。充足可能な場合、当該プロシージャーは、当該状態がｑ＿｛ｎ＋ｉ｝も充足するかどうか検査する（第７行）。ｑ＿｛ｎ＋ｉ｝も同時に充足する方法がない場合は、証拠が検出されたとみなされる（第８行）。充足可能な場合は、ｐ＿ｉとｑ＿｛ｎ＋ｉ｝の両方を充足する状態が考慮対象から除外される。これはｗｈｉｌｅループによって行われる（第１１～１７行）。除外集合Ｃ’（初期化により空にされる）は、ループの各反復において、充足する状態キューブ（すなわち、ｐ＿ｉとｑ＿｛ｎ＋ｉ｝の両方を充足する状態）を追加することによって更新される（第１６行）。次に、Ｃ’以外にｐ＿ｉを充足する状態があるかどうかが検査される（第１３行）。該当する状態が存在しない場合には、当該プロシージャーはｗｈｉｌｅループから分離し、ｉを漸増させながらさらに長い証拠を探索しなければならない（第１４行）。一方、このような状態が存在する場合は、その状態がｑ＿｛ｎ＋ｉ｝も併せて充足するかどうかが検査される（第１５行）。充足しない場合は、証拠が検出されたとみなされる。充足する場合は、ｗｈｉｌｅループが実行され、これらの状態が除外される。完全性については、当該プロシージャーをループなしのスケルトンのみに絞り込んで実行するかどうかを選択できる（第２０～２４行）。

【００５７】

ここで、除外集合内の（ｐ＿ｉ　＆　ｑ＿｛ｎ＋ｉ｝）の解を漸増的に列挙するために、ＳＡＴソルバーが使用されていることに注目する必要がある。ＳＡＴとＢＤＤの組み合わせは、イメージ集合の計算にも使用できる（非特許文献１７参照）ほか、ＳＡＴベースのＢＭＣフレームワーク内では制約として使用することができる。

【００５８】

我々が遭遇したもう１つの有用なＣＴＬ特性は、ＡＧ（ｐ　－＞ＥＦ　ｑ）である。これはリセット可能性、デッドロックの不在などの検査に使用できる。ＳＡＴベースのＢＭＣフレームワーク内では、改変が行われることと、ネストされたＥＦ演算子が不動点の性質を有することを理由として、この特性を検査することは不可能である。ただし、実用使用では、ｑが真の場合において対象となる状態の最終的な到達可能性を検査する際には、設

計者は常に境界を意識しているものである。このようなケースでは、ｎが漸増されるＥＸ：ｎ演算子は、ＥＦ演算子の有界近似としての役割を果たす。

【００５９】

実用使用では、正当性特性の範囲を拡張して、ＣＴＬのすべてのモダリティを含めることができる。ここで注意を要するのは、改変にいかに対処するかという問題と、（証拠パスではなく）証拠グラフをいかにして継続的に追跡するかという問題である。概してこれには、ＱＢＦソルバーか、ＳＡＴとＢＤＤとの組み合わせを使用することが必要とされる。

【００６０】

２．　漸増的ＳＡＴ手法

上述したように、ＢＭＣには、境界ｋを漸増させながら、問題に対応する一連のＳＡＴインスタンスを解決する作業が伴う。複数のＳＡＴインスタンスが各々の文節集合の間に空でない交差部分を有する場合には、これらのインスタンスの間で対立文節を共有でき、この共有によりＳＡＴの解決時間が全体的に短縮できることが、数人の研究者（非特許文献１３、１９参照）によって観察されている。特に、ＢＭＣ問題のｋインスタンスとｋ＋１インスタンスは遷移関係のｋ回の展開を共有するため、回路制約の重複が多く発生する。問題のｋインスタンスの検査中に演繹された制約を再使用する制約共有手法（非特許文献１３、１４参照）は、この事実を利用したものであり、これにより全体的な検証時間が短縮されるという効果が得られている。

【００６１】

本発明で考察する実施例のＢＭＣ実装では、共有は、遷移関係の展開に伴って回路制約の間で発生するだけでなく、特性の変換から生じる制約の間でも発生する。さらに、制約を漸増的に追加する本発明の変換スキーマでは、境界ｋを漸増させたときに複数のＳＡＴ問題が生成され、分割が使用されたときには単一のｋインスタンス内でさえも複数のＳＡＴ問題が生成される（カスタマイズ特性変換の記載を参照）。複数のＳＡＴインスタンス間で制約を共有することに重点を置くことにより、漸増的ＳＡＴ手法によって全体的な検証時間が短縮される可能性が生まれる。

【００６２】

ＳＡＴソルバーの基本的な制約共有手法は、以下のように実行される（非特許文献１４参照）。２つのＳＡＴインスタンスをＳ＿１およびＳ＿２とすると、共通する文節の集合Ｙ＿０のみから演繹された対立文節（すなわち、Ｓ＿１とＳ＿２の両方に存在する文節）が識別される。識別は、Ｙ＿０文節の最初のマーキングに基づいて行われる。続いて、コンフリクト分析によって生成された各々の対立文節を対象にマーキングを実行する。１つの対立文節において、対立につながるすべての文節がマーキングされた場合は、その対立文節もマーキングされる。

【００６３】

本実施例の実装では、各文節は「ｇｆｌａｇ」と呼ばれるビット・ベクトル・フィールドを有する。ｇｆｌａｇの各ビットは、その文節がそのビット位置に対応するグループに属するかどうかを示す。さらに、各文節は、「制約文節」、「回路文節」、「対立文節」という３つのタイプに分類される。１つの制約文節は、最大１つのグループに属する。回路文節は、どのグループにも属さない。コンフリクト分析時に追加された対立文節は、１つのグループに属し、かつ対立につながる文節が存在する場合に、そのグループのメンバーとなる。回路文節のみから演繹された対立文節（初期状態制約を含む）は、必ずしも複製できるとは限らないが、再使用は常に可能である。

【００６４】

ここで、ＥＦ（ｐ）特性の証明に漸増的ＳＡＴ手法を使用した簡単な例を示す。ｉ番目の時間フレームにおいて、制約文節（ｐ＿ｉ＝１）が新規グループｇｉｄに追加される。（前述したように、ｐ＿ｉは、ｉ番目の時間フレームにおける特性ノードｐである）。結果がＵＮＳＡＴになると、この文節グループｇｉｄは、以降の時間フレームで再使用されないので削除される。この削除に伴って、グループｇｉｄのすべての対立文節と制約文節が除去される。ただし、回路文節のみから演繹された対立文節はそのまま保持される。これ

らの回路文節は、時間フレーム間で共有することができる。さらに、結果がＵＮＳＡＴとなったため、グローバル学習を使用して、その文節（ｐ＿ｉ＝０）を文節グループｒｏｏｔ＿ｇｉｄに追加する。グローバル学習されたこれらの文節から演繹された対立文節もまた、以降の時間フレームで再使用できる。以下に示すのは、漸増的ＳＡＴ手法の使用法を示す疑似コードである。

【００６５】

```
// N=Max depth
// p_i=property node at i-th time-frame
EF(p)
{
// allocate a group for learned clauses
root_gid=alloc_group_id();
for(i=0;i<N;i++){
// allocate a new group id
gid=alloc_group_id();
// add the(p_i=1) as constraint clause to
// the group, gid.
add_constraint(p_i, gid);
status=sat_solve();
if (status==SAT) return true;
// delete conflict and constraint clauses
// that belong to the group gid. Note that
// conflict clauses derived only from the
// circuit clauses are never deleted.
delete_group_id(gid);
// add the learned clause (p_i=0) to the
// group root_gid.
Add_constraint(!p_i, root_gid);
}
}
```

【００６６】

他の特性の変換においても、対立文節と学習された制約文節に対して漸増的ＳＡＴ手法を使用するために、同様の変更が行われる。

【００６７】

３．　回路の単純化

本発明で考察するＢＭＣ実装の実施例においては、前処理段階と、設計の遷移関係を展開する際の特性検査の段階で、回路単純化手法が使用される。これは、生成されるＳＡＴ問題を単純化して、全体的な検証時間を短縮することを目的とする。さらに、回路単純化手法は、ＣＮＦベースのＳＡＴ決定プロシージャー内に定数を伝搬する方法に比較して、定数の処理を効率的に行えることが確認されている。こうした定数は、フリップフロップ上の定数値を伴う初期状態制約と環境的制約、および特性検査時に追加された学習済み定数制約が原因で発生する。

【００６８】

回路の単純化は、非標準形の２入力ＡＮＤ／ＩＮＶＥＲＴＥＲグラフ表現（非特許文献２０参照）を使用し、この種のグラフ表現に対してオンザフライ式のリダクション・アルゴリズム（非特許文献２１、２２参照）を実行することによって実現される。このグラフは

、設計と、複数の時間フレームにまたがる記号的計算において算出されるブール関数の両方を表現するために使用される。この方法の不利な点は、標準形のＢＤＤとは異なり、ＡＮＤ／ＩＮＶＥＲＴＥＲグラフは非標準形であることである。この方法の有利な点は、このグラフはＢＤＤに比較して、サイズが特定の関数や変数の順序の影響をはるかに受けにくく、簡潔なことである。

【００６９】

オンザフライ式リダクション・アルゴリズムは、ＡＮＤ／ＩＮＶＥＲＴＥＲグラフを表現するための効率的な関数ハッシング・スキームをベースとする。ハッシュ・テーブルは、ＢＤＤと同様に、構築時に構造的な冗長性を除去するために使用される。さらに、ローカルの４入力副構造をすべて標準形の表現に変換する構造的な２階層ルックアップ・スキームが適用され、ローカルの冗長性が効果的に除去される。ローカルの２階層ルックアップが適用されない場合は、単純書き換え方式によって、回路グラフがさらに縮小される（詳細については非特許文献２２を参照）。具体的には、これは、多数の同形の副構造を識別し、これらの副構造を同一のサブグラフにマッピングすることによって、グラフを大幅に圧縮する方式である。直感的に表現するならば、この単純化グラフによりＳＡＴソルバーの探索空間が縮小され、ひいてはブール推論の効率が高まる。有界到達可能性分析への応用（非特許文献１２参照）からも明らかなように、こうした単純化によりＳＡＴにもたらされる計算上の優位性はきわめて顕著である。

【００７０】

４．　ハイブリッドＳＡＴソルバー

本発明におけるＢＭＣエンジンの実施例では、ハイブリッドＳＡＴソルバーが使用されている。このハイブリッドＳＡＴソルバーは、元来の回路形式の論理式と、ＣＮＦにおける学習された対立文節とをそれぞれ処理する一方で、決定変数の選択、ＢＣＰ、およびバックトラッキングにおいて革新的な最新鋭機能を採用している。ここでは特に、回路ベースの方式とＣＮＦベースの方式の主要ステップにおける重要な違いと、ハイブリッド方式とすることで両方式からどのような利点が実現されるかについて説明する。

【００７１】

ａ）　回路ベースＢＣＰの最新技術の現状

ＳＡＴソルバーの一部を成すＢＣＰは、多数のインスタンスにおいて総実行時間の約８０％を占めるとされる。これは、ＢＣＰをどんな形であれ改良できれば、ＳＡＴソルバーの全体的なパフォーマンスが大幅に向上することを意味する。

【００７２】

従来の回路をベースとするブール推論実装（非特許文献２２から２５参照）は、ＩＮＶＥＲＴＥＲを別個の頂点またはゲート入力上の属性として備えた、ＡＮＤおよびＯＲのゲート頂点をベースとする表現を使用する。ＡＮＤおよびＯＲゲートにまたがって定数を伝搬する方法はよく知られているが、速度は実装によって大きく異なる傾向がある。一例として非特許文献２２を取り上げると、この技術は含意の伝搬を高速化する手段としてルックアップ・テーブルを使用している。このルックアップ・テーブルは、頂点の入力と出力の現在値に基づいて、暗黙値をカプセル化するゲートの次の「状態」と、その頂点に対して採るべき次の処置を決定する。図３に示す非特許文献２２の総称頂点タイプ用のアルゴリズムｉｍｐｌｙは、回路グラフ上で繰り返し実行される。

【００７３】

このアルゴリズムは、各頂点について、新しい暗黙値と以降の処理の方向を決定する。例として、図４（出典は非特許文献２２）に、２入力ＡＮＤゲートの含意ルックアップ・テーブルに含まれるいくつかのケースを示す。ブール論理の場合、ｊｕｓｔｉｆｉｃａｔｉｏｎ＿ｑｕｅｕｅにおいて新たなケース分割をスケジュールする必要があるのは、１つのケース（ＡＮＤ頂点の出力における論理０）だけである。他のケースはすべて、対立を発生させるか（この場合、アルゴリズムはバックトラッキングのために戻る）、さらなる含意を発生させるか、ｊｕｓｔｉｆｉｃａｔｉｏｎ＿ｑｕｅｕｅの次の要素を処理するための戻りを発生させるかのいずれかである。この含意アルゴリズムは、オーバーヘッドが少

ないため非常に効率的である。目安として示すと、２５６　ＭＢのメモリを搭載した７５０　ＭＨｚのＩｎｔｅｌ　ＰＩＩＩでは、毎秒１００万含意以上を実行できる。

【００７４】

１．　ＣＮＦベースＢＣＰとの比較

Ｃｈａｆｆ（非特許文献１０参照）に示されるＣＮＦベースＢＣＰでは、２リテラル監視と遅延更新によって効率化が図られている。この方式は、サイズの大きい文節が不必要に走査される状況を回避できるため、文節が大きい場合に際だって有利である。また、この方式では、オーバーヘッドの削減を狙って、充足された文節の追跡は行われない。ただし、充足された文節を訪れることにより必然的に生じるコストまでは解消されない。具体的には、文節が監視対象ではないリテラルに割り当てられたことにより充足された場合でも、監視対象のリテラルへのポインターは更新される。これに加えて、ゲートを文節に変換するプロセスに組み込まれる固有のオーバーヘッドもある。回路ベースの方式では１つのゲートはモノリシックなエンティティであるのに対し、ＣＮＦベースの方式では、１つの２入力ゲートが３個の文節に変換される。そのため、回路方式では、１ゲートに存在する１含意について図４のテーブルを１回探索するだけでよいが、ＣＮＦ方式では、複数の文節を処理する必要がある。

【００７５】

回路ゲートから生じる文節は概して短く、これらの文節に関しては、上記の相違点がＢＣＰ時間にかなり大きな違いをもたらす。次項に示すように、ゲート表現を対象とするＢＣＰは、Ｃｈａｆｆのような最新鋭のＣＮＦベース・ソルバーのＢＣＰに比較して、一貫してはるかに高速である。

【００７６】

ｂ）　対立ベースの学習の効果を活用するハイブリッド方式

一方、対立ベースの学習済み文節のようにサイズの大きい文節の場合は、回路ベースのＢＣＰにゲート・ツリーとして付加すると、学習済み構造の規模を過剰に増大させる恐れがある。学習済み構造のサイズを過剰に増大させるゲート・ツリーを付加することは、回路サイズの大幅な増大につながる。これはさらに、含意数を増大させ、ひいては回路構造にＢＣＰを実行したことによる利得を相殺する結果となる。サイズの大きい文節の場合は、これらをモノリシックな文節として維持し、ＣＮＦベースの２リテラル監視と遅延更新の利点を活用して処理した方が効率的である。

【００７７】

次に、上記の観察を踏まえて、ハイブリッド方式について考察する。このハイブリッド方式とは、均質ゲートのデータ構造を使用して回路ベースの論理表現を維持する一方で、状況に応じて、学習された文節をＣＮＦとして維持し、別個に処理する方式である。

【００７８】

ｃ）　ＢＣＰの結果

表１（図５）に示す時間は、２５６　ＭＢのメモリを搭載した７５０　ＭＨｚのＩｎｔｅｌ　ＰＩＩＩ上で１００万含意の処理に要した秒単位のＢＣＰ時間である。サンプルとしては、大規模な工業用回路にＢＭＣを適用して導出したサイズの大きい論理式を使用した。「ハイブリッド」と「Ｃｈａｆｆ」の欄に示した時間は、ハイブリッド方式とＣＮＦ方式を完全に同一の論理式を対象に実行した結果である。ＢＣＰ時間には、ＳＡＴプロセス中に付加された学習済み文節のＢＣＰに要した時間と、最初のゲート文節のＢＣＰに要した時間が含まれる。１次入力とゲートの数で表した式のサイズは、「１次入力」と「ゲート」の欄に示されている。「ＣＨ」の欄は、ＣＮＦのＢＣＰ時間とハイブリッドのＢＣＰ時間との対比を比率として示したものである。この表から、サイズの大きい式においては、ハイブリッド方式はＣｈａｆｆのＣＮＦベース方式に比較して、一貫して高速である。

【００７９】

学習済み文節のＢＣＰで達成されるオーバーヘッドの削減を実証するため、１００万含意当たりのＢＣＰ時間を、同じ式に含まれるゲート文節のみを対象とした回路ベース方式との対比として示した。これは、「構造」欄に示している。「ＣＳ（ＨＳ）」欄は、「Ｃｈ

ａｆｆ（ハイブリッド）」の時間と「構造」の時間との対比を比率として示したものである。これらの欄（「ＣＳ」および「ＨＳ」）により、ゲート文節のみを対象とした場合のＢＣＰ時間と、ゲート文節と学習済み文節を対象とした場合のＢＣＰ時間とを比較することが可能になる。この比較から、サイズの大きい学習済み文節がＢＣＰ時間のオーバーヘッドを大幅に増加させているのは明らかである。

【００８０】

ｄ）　ハイブリッドＳＡＴソルバーの総合的な利点

これまでＢＣＰが高速であることを実証してきたが、これだけではハイブリッドＳＡＴソルバーの利点を語り尽くしたとは言えない。そのためには、ハイブリッド方式によってＳＡＴプロセス全体が高速になることを実証する必要がある。ハイブリッド方式を使用すると、純粋なＣＮＦ方式では使用できない多数の新たな回路ベース・ヒューリスティックと利点を活用する道が開かれる。この項では、ハイブリッド方式によりもたらされる利点について考察する。この比較のため、表２（図６）にＳＡＴ実行時間を示す。ここに示す時間は、２５６　ＭＢのメモリを搭載した７５０　ＭＨｚのＩｎｔｅｌ　ＰＩＩＩ上での実行時間である。論理式は、３個の大規模な工業用回路（バス、アービター、コントローラー）と、非特許文献２６に掲載されるいくつかのパブリック・ドメインのベンチマークに対してＢＭＣを適用して導出したものである。結果を適正なものに保つため、ハイブリッド方式を７０以上の論理式に対して実行し、その中から４０秒を上回るＣＰＵ時間を要した論理式の結果だけを報告する。これらの論理式を、充足不能なインスタンスと充足可能なインスタンスに分配した。論理式の式サイズは、１次入力とゲートの数を示す「１次入力」欄と「ゲート」欄から判断できる。

【００８１】

ｅ）　同一のヒューリスティックにおけるＣｈａｆｆとハイブリッドの比較

第１の比較では、ハイブリッド・ソルバーとＣｈａｆｆを完全に同一のヒューリスティックに適用した。すなわち、この２つは、ＢＣＰの相違を除いてまったく同一のアルゴリズムを使用して、含意の処理、対立ベースの学習、バックトラッキング、決定変数の選択の順に実行された。ヒューリスティックは同一とはいえ、複数のノードが対立した際の対立ノードの選択は制御されなかったので、多少の相違が存在することは否めない。充足不能なインスタンスについては、常に探索空間全体を走査する必要があるので上記の相違による影響はほとんどないが、充足可能インスタンスについては、２つのソルバーのうち１つが偶然に早期に解決されるインスタンスに遭遇する状況が発生するため、顕著な影響が出る可能性がある。この事実を考慮すると、この制御された実験おいては、充足不能なインスタンスのみを妥当なデータとみなすべきであろう。「Ｃｈａｆｆ」欄と「Ｈ」欄は、表２（図６）のＣｈａｆｆソルバーとハイブリッド・ソルバーの時間を示す。ハイブリッド・ソルバーの全体的なパフォーマンスは、Ｃｈａｆｆのそれよりもはるかに良好であることは明らかである。ハイブリッドの時間に対するＣｈａｆｆの時間の典型的な比率は１．３以上であり、最大のケースでは３．７５にも達する。充足不能な全インスタンスにおいてハイブリッド・ソルバーが費やした総時間に対するＣｈａｆｆが費やした総時間の比率は、１．４８である。表２（図６）に示す充足可能なインスタンスについては、ハイブリッドに対するＣｈａｆｆの比率（Ｃｈａｆｆ／Ｈ）は１．０を中心として均等に分布しているが、標準偏差は大きいという、予想通りの結果となっている。

【００８２】

５．　回路ベースのＳＡＴヒューリスティック

この項では、ハイブリッド・ソルバーの機能強化に使用される回路ベース・ヒューリスティックの詳細を示す。ここでは、表３（図７）を参照する。表３（図６）で使用するサンプルは、表２（図６）と同じである。「Ｈ１」欄は、前項で説明したＣｈａｆｆと同じヒューリスティックを使用した場合のハイブリッド・ソルバーの実行時間を示す。

【００８３】

ａ）　含意を追跡する順序

Ｃｈａｆｆでは、含意はＦＩＦＯメカニズムを使用して追跡される。すなわち、基本的に

は、含意は生成された順序で処理される。ハイブリッド方式では、ゲートのファンアウトと方向性の情報に基づき、回路パスに沿って含意を追跡することができる。学習済み文節から生成された含意をパスに沿って追跡する一方で、ゲートから生成された含意はＦＩＦＯ方式で追跡する方法が非常に効果的であることが判明した。「Ｈ２」欄は、このヒューリスティックを使用した場合の総実行時間を示す。この欄の結果から、このヒューリスティックでは、ほぼすべてのサンプルにおいて（１８中１４）、Ｈ１を上回る速度が達成されることは明らかである。「Ｈ２」の右側にあるすべての欄では、このヒューリスティックが使用されている。

【００８４】

ｂ）　分岐変数決定戦略

決定戦略では、分岐点となる未割り当ての変数と値が選択される。これまで数種類の戦略が提唱されているが、決定的な勝者はまだ出現していない。最も大きな成功を収めた戦略はいずれも、何らかの形式の動的リテラル・カウントをベースとしている。次に、回路情報を使ってこの基本メカニズムの機能を強化する方法について考察する。

【００８５】

ＣＮＦベース・ソルバーは、最多の文節を充足するリテラルを選択するために、文節に含まれる正と負のリテラル数をカウントする。ここで重要なのは、どうしたら各ゲートの入力と出力において無矛盾値を取得できるかという問題である。図８に示すように、文節に含まれるリテラルをカウントするだけではこの問題に対処することはできない。図８（ａ）を参照すると、ＯＲゲートｏは入力ａおよびｂを有し、ファンアウト・ゲートはｘ、ｙ、およびｚを有している。図８（ｂ）は、生成された文節を示す。この図から、変数が入力であるとき、文節内のその正と負のリテラル数は互いに相殺されることが分かる。また、変数がゲートの出力のときには、正のリテラル数が負のリテラル数よりも常に１多くなる。図８（ｂ）の対応するスコアは、それぞれ５と４である。端的に言うと、リテラル数からは、ゲートから生成された文節に関する有益な情報は得られない。一方、ゲートのファンアウト情報からは、意思決定に必要とされるゲートの正と負のファンアウト数を正確に決定できる。これは、「ファンアウト・ヒューリスティック」と呼ばれる。「Ｈ２－ｆｓ」欄は、このヒューリスティックでのＳＡＴ時間を示す。このヒューリスティックが決定的な勝者というわけではないが、過半数を少し上回るサンプル（２６／４５）において、原型ハイブリッド方式（Ｈ２）よりも良好な結果をもたらすことは特筆に値する。

【００８６】

式の充足性を決定しない文節は、「非アクティブな文節」と呼ばれる（非特許文献２７参照）。非アクティブな文節は、観察不能になったゲートから生じる。これらの文節とそれに含まれる変数の決定を処理することは、基本的には無駄な努力である。これらの文節を動的に検出して除去するためには、文節にマーキングとアンマーキングを繰り返し行う必要がある。これらの実行時間演算を行うと枝取りされた探索空間が得られるが、マーキングとアンマーキングを繰り返し実行することで全体的なパフォーマンスに損失が生じる。このパフォーマンス向上は、到達可能性分析にのみ見られることに注意する必要がある。回路に対して演算を行う伝搬－位置調整型のブール推論メカニズム（非特許文献２４参照）では、観察不能なゲートと非アクティブ領域の動的検出は自動的に開始される。位置調整を必要とする変数は、「フロンティア変数」と呼ばれる。この戦略は、変数決定の対象を動的に変動するフロンティア上の未割り当てゲートに絞り込むことによって適用される。これは、「フロンティア・ヒューリスティック」と呼ばれる。このヒューリスティックにはさらに、充足する割り当てが存在するときには、より迅速にそれに到達できるという利点もある。「Ｈ２－ｆｔ」欄は、このヒューリスティックでのＳＡＴ時間を示す。サンプルの８０％において原型ハイブリッド方式（Ｈ２）よりも良好な結果が得られた。

【００８７】

「Ｈ２－ｆｔ－ｆｓ」欄は、フロンティア・ヒューリスティックとファンアウト・ヒューリスティックの両方を使用した場合の実行時間を示す。ここでも、サンプルの約７３％においてＨ２を上回る速度が観察された。

【００８８】
ｃ）　ＸＯＲ／ＸＮＯＲゲートの学習
多数のＸＯＲおよびＸＮＯＲゲートを有する回路では、ＳＡＴソルバーのパフォーマンスが低下することが知られている。均質なゲートを有する回路表現を与えられた場合は、回路内のＸＯＲおよびＸＮＯＲゲートを抽出することは可能であり、かなりの効率も期待できる。ハイブリッド表現では、これらのＸＯＲ／ＸＮＯＲゲートの文節が学習され、ＣＮＦ文節データベースに加えられる。「Ｈ２－ｆｔ－ｆｓ－ｓｉｘ」欄は、この学習をフロンティアおよびファンアウト・ヒューリスティックと共に適用した場合の実行時間を示す。これら３つのヒューリスティックをすべて採り入れた場合、サンプルの６２％以上で基本的なハイブリッド方式（Ｈ２）を上回る速度が観察された。

【００８９】
ｄ）　ハイブリッド・ソルバーでの最良の結果
「Ｈ－ｂｅｓｔ」欄に、ハイブリッド・ソルバーで得られた最良の結果を示す。すべてのサンプルにおいてハイブリッド・ソルバーがＣｈａｆｆよりも高速なことは明らかである。「Ｃｈａｆｆ／Ｈ－ｂｅｓｔ」欄に、最良のハイブリッド・ソルバー時間に対するＣｈａｆｆ時間の比率を示す。すべてのサンプルで５０％以上の速度向上が見られ、サンプルの６９％で２を上回った。表３（図６）のヒューリスティックは明らかに未成熟であり、さらなる研究を要する。とはいえ、Ｈ２－ｆｔヒューリスティックはきわめて良好に動作するので、デフォルトとして使用できるのは明らかである。

【００９０】
ｅ）　有界モデル検査における含意
ＳＡＴはＢＭＣのようなアプリケーションにおいてコア・エンジンとなる。事実、５０～１００時間フレームにわたる回路分析を必要とする典型的なＢＭＣ処理では、ＳＡＴソルバーが１回の実行で数千回も呼び出されることが予想される。ＳＡＴソルバーは１回の呼び出しにつき数分間程度を消費するため、長いＢＭＣ実行の場合は完了までに数日間を要することになる。そのため、コア・エンジンで２倍の速度向上が実現できることは、絶対的な実行時間が大幅に短縮されることであり、その影響はきわめて大きいものとなる。本研究のもう１つのテーマは、回路ＳＡＴ手法は有力な候補であり、ＣＮＦベースの手法に勝ることを明確に実証することである。実用使用における効果としては、ＳＡＴのために回路全体をＣＮＦデータ構造にコピーしなくてもよいため、それに伴うオーバーヘッドを見込む必要がないことが挙げられる。これは、これらのアプリケーションのメモリ所要量がほぼ半減されるため、より大規模な回路にスケーリングすることが可能になり、ＢＭＣの場合はスケーリングできる時間フレーム数も増大する、という利点をもたらす。

【００９１】
６．　生産設計に関する実験結果
生産設計を検証するため、ＢＭＣエンジンの実装をＤｉＶｅｒ内で適用した。各検証タスクでは、対象となる特性の証拠（または反証）を探索した。これらの特性は、安全性と活性とした。結果を表４（図９）にまとめる。ほとんどの実験は、２５６　ＭＢのメモリを搭載した７５０　ＭＨｚのＩｎｔｅｌ　ＰＩＩＩ上で実施した。表内に「（Ｓ）」を付記して示した一部の実験は、Ｓｕｎ　ＵｌｔｒａＳｐａｒｃ　４４０　ＭＨｚ、１ＧＢのワークステーション上で実施した。

【００９２】
各設計を対象に、第４欄に示す５つのオプション・セットに関して実験を行った。最初の４つのオプション・セットは、カスタマイズ変換の４種類の組み合わせ（「＋／－　Ｔ」として示す）と、構造的同形を使用した回路単純化（「＋／－　Ｃ」として示す）に関連する。（定数の伝搬により実現された回路単純化を、常にデフォルトとして使用する。）第１のオプション・セットｏｐ１はこのいずれも使用せず、第２のオプション・セットｏｐ２は回路単純化のみを使用し、第３のオプション・セットｏｐ３はカスタマイズ変換のみを使用し、第４のオプション・セットはこの両方を使用する。この４つのオプション・セットはすべて、第６項で説明した漸増的ＳＡＴ手法を実行する能力を有するＣｈａｆｆ

ＳＡＴソルバー（非特許文献１０参照）の修正版を使用する。最後に、第５のオプション・セットｏｐ５も、カスタマイズ変換と回路単純化の両方を使用する。これは、第８および９項で説明したハイブリッドＳＡＴソルバー（「＋Ｈ」として示す）を使用することを除いて、第４のオプション・セットと同じである。

【００９３】

漸増的ＳＡＴ手法の使用により実現可能なパフォーマンスの利得はどの程度かを調べることも興味深い。そのため全５オプション・セットについて、ＳＡＴソルバーで漸増的ＳＡＴ手法を使用する場合と使用しない場合の実験も行った。カスタマイズ変換は、ＳＡＴソルバーで漸増的ＳＡＴ手法の使用の有無にかかわらず、常に漸増的な公式化を使用することに留意されたい。

【００９４】

表４（図９）に、いくつかの設計および特性に関する全５オプション・セット（１～５を付番）の結果を示す。２番目と３番目の欄はそれぞれ、各特性に関する影響の静的コーン（全オプション・セット間で同一）内に存在するフリップフロップ数とゲート数を示す。５番目の欄は、証拠／反証を検出するために展開された時間フレーム数（すなわち、ＢＭＣのパラメータｋ）を示し、６番目の欄は、そのｋにおいて証拠が検出されたかどうかを示す。次の２欄は、漸増的ＳＡＴ手法を使用した場合と使用しなかった場合（それぞれ、「＋Ｉｎｃ」および「－Ｉｎｃ」として示す）にかかった時間（単位：　秒）を示す。最後の欄は、漸増的　ＳＡＴ手法を使用した検証を実行するために必要とされたメモリ量（単位：　メガバイト）を示す。（ほとんどのサンプルでは、漸増的ＳＡＴを使用しない検証で消費されたメモリ量も同程度であった。）

【００９５】

オプション・セット２～５では基本ＢＭＣ（オプション・セット１）に比較して一貫してパフォーマンスが向上したことから、本発明の機能拡張の効果は明らかである。サイズが大きいサンプルは、同じ境界ｋに関して最大２オーダーのパフォーマンス向上を示した。基本ＢＭＣは、割り当てられた時間（１０，０００秒）内に、機能拡張を行ったＢＭＣよりも少ない数の時間フレームしか完了できないケースが多数あったことは注目に値する。特に、バスのサンプルでは、基本ＢＭＣはこの時間内ではついに証拠を検出できなかった。

【００９６】

ほとんどのサンプルにおいて、カスタマイズ変換のみを使用した機能拡張（ｏｐ３）は、回路単純化のみを使用した機能拡張（ｏｐ２）よりも良好なパフォーマンスを示した。また、この両方を組み合わせた機能拡張（ｏｐ４）は、いずれか一方のみを使用した機能拡張に比較して常に良好な結果を示した。さらに、ハイブリッドＳＡＴソルバーを使用した機能拡張（ｏｐ５）は、ほとんどのサンプルにおいてこれよりもさらに高いパフォーマンスを示した。

【００９７】

漸増的ＳＡＴ手法の使用については、ＢＭＣの異なるｋインスタンス間で回路制約だけが共有される基本ＢＭＣ（ｏｐ１）において、漸増的ＳＡＴ手法の使用による利得が最も少ないことに注目する必要がある。漸増的ＳＡＴの真の利点は、上記に加えてＢＭＣの各ｋインスタンス内で制約を共有することも可能な本発明のカスタマイズ変換に見ることができる。カスタマイズ変換を使用する場合（ｏｐ３、ｏｐ４、ｏｐ５）の改善率は、使用しない場合（ｏｐ１、ｏｐ２）よりも概して高い。これは、ＳＡＴ解決時間が大きな影響を及ぼす比較的大きなサンプル（Ｂｕｓ３、Ｄｍａ２、およびＤ１－Ｐ５）の結果から明白である。さらに、ＳＡＴソルバーで漸増的ＳＡＴ手法を使用することによる利得は、カスタマイズ変換を使用する場合よりも少ない。これは、ＳＡＴソルバーが制約の共有を行わない場合でさえも、分割戦略として漸増的な公式化を採ることに大きな利点があることを示している。

【００９８】

７．　帰納による安全性特性の証明

ＢＭＣエンジンは、有界長の反証を検出することに加えて、帰納証明の実行にも使用できる（非特許文献１６参照）。深度ｋを増大させて実行される帰納と、ループなしパスに対する制限を組み合わせると、安全特性の完全な証明手法が得られる（非特許文献５、１１参照）。深度ｋの帰納は、以下の２ステップから成る。

【００９９】

基本ケース：その特性が、初期状態から始まる長さｋの各パス上で有効であることを証明する。

帰納ステップ：その特性が、任意の状態から始まる長さｋのパス上で有効であり、さらには、長さ（ｋ＋１）のパスの拡張上でも有効であることを証明する。

【０１００】

ループなしパスに限定することにより、「パス内に同一の状態がない」という追加の制約が課される。ここで、基本ケースでは初期状態制約が使用されるが、帰納ステップではこの制約は使用されないことに注意する必要がある。したがって、帰納ステップには到達不能な状態も含まれる可能性がある。実用使用では、追加の制約（すなわち、特性自体よりも強力な帰納不変式）を使わずに帰納証明を完了することは不可能なこともある。この場合は、到達可能性の制約など、設計者に既知の回路制約を使用して帰納不変式を強化することができる。

【０１０１】

ａ）　帰納証明のためのＢＭＣプロシージャー

この項では、ＢＭＣエンジンで使用される帰納プロシージャーについて解説する。以下に擬似コードを例として示す。

【０１０２】

```
/* p_i, c_i : property node p, constraint node c
at i-th time frame,
starting from unconstrained state
I: initial state constraint
R: Reachability constraint,
e.g. over-approximate reachable set */
1  bmc_induction_proof(p, c, R, I)
2  {
3    BC=I;
4    IC=R & c_0;
5    if (!is_sat(IC & !p_0)
6        return true;
7    for (i=0;i<N;i++) {
8      BC=BC & c_i;
9      // Base case: if sat, counterexsample
10       if (is_sat(BC & !p_i))
11           return false;
12       BC=BC & p_i;
13       IC=IC & p_i;
14       IC=IC & c_i+1;
15       // Inductive step: if unsat, found proof
16       if (!is_sat(IC & loop_free(0, i+1) & !p_i+1))
17           return true;
18   }
```

１９　ｒｅｔｕｒｎ　ｉｎｃｏｎｃｌｕｓｉｖｅ；
２０　｝
【０１０３】
ここで、ｐは、真であることを証明すべき安全性特性に対応するモニター述語である。すなわち、正当性特性はＧ（ｐ）である。ＬＴＬで表現された任意の安全性特性を与えられると、質問票作成手法（非特許文献１参照）を使用して、「安全性特性はモニター述語がすべての到達可能な状態において真である場合にのみ真である」というモニター述語を得ることができる。ここに示した擬似コードでは、ｃは真であることが既知の回路制約、Ｉは初期状態制約、Ｐは到達可能性制約をそれぞれ示す。本書で解説する証明プロシージャーのサンプルは、ｃとＲは有効であることを前提とする。すなわち、ｃ＿ｉはすべての時間フレームにおいて真でなければならず、Ｒは到達可能状態の過剰近似でなければならない。（これが得られない場合は、各々が擬似コード内で「真」であるとみなすことができる。）
【０１０４】
擬似コードでは、ＢＣは基本ケースの検査を実行するための文節データベース、ＩＣは帰納ステップ検査を実行するための文節データベースである。このプロシージャーは、初期化によって開始される（第３、４行）。第１のＳＡＴ検査（第５行）は、ｃとＲを充足する状態が！ｐも同時に充足するかどうかを確認するために実行される。このような状態が存在しない場合、特性ｐは明らかに真である。これは、帰納証明がまだ開始されない段階の早期終了ケースとされる。ｉ上のループ（第７～１８行）は、深度をユーザー定義の限界値Ｎまで増大させて帰納証明を実行する。最初に、基本ケースが検査される。これは、制約ｃ＿ｉを追加し、ｉ番目の時間フレームにおける！ｐ＿ｉの充足可能性をチェックすることにより行われる。充足可能な場合は、初期状態から始まる反証が検出されたことを意味するため、特性は偽である（第１０、１１行）。一方、充足可能でない場合は、ｐ＿ｉが学習され、ＢＣデータベースに追加される（第１２行）。次に、（ｉ＋１）番目の時間フレームにおける帰納ステップが検査される。ここでは、帰納的仮説によりｐ＿ｉは真であると想定され（第１３行）、制約ｃ＿ｉ＋１が追加され（第１４行）、さらには、長さ（ｉ＋１）のパスはループなしという制限のもとで！ｐ＿ｉ＋１の充足可能性が検査される（第１６行）。これが充足可能でない場合は、帰納証明が成功したのであり、特性はまさしく真である。一方、充足可能な場合は、帰納証明は深度ｉで失敗したのであり、引き続きｉを漸増させてループを再試行する。そのまま限界値Ｎに到達すると、結果は不確定となる。
【０１０５】
このプロシージャーは、基本ケースと深度ｉにおける帰納ステップについてそれぞれモノリシックＳＡＴ式を生成するのではなく、充足不能なＳＡＴインスタンスから学習しながらＳＡＴ下位問題を漸増的に構築する。数個の工業用設計で得られた実験結果により実証されたように、上記の機能拡張で強化した本発明のＢＭＣエンジンは、実用使用においてこうした証明を効果的に実行することができる。これらの結果は、４４０　ＭＨｚ、１ＧＢのＳｕｎ　ＵｌｔｒａＳｐａｒｃワークステーション上で実施した実験の結果をまとめた表５（図１０）に示されている。
【０１０６】
ＢＭＣエンジンが反証を検出できなかった設計（第１欄）と安全特性（第２欄）に対しては、帰納証明を実行した。これらの証明では設計者から提供された回路制約を使用したが、到達可能性制約は使用しなかった。第３欄は、特性に関する影響のコーン内に存在するフリップフロップ数（＃ＦＦ）とゲート数（＃Ｇ）を示す。第４欄は、検証状況を示す。ここで、「Ｔ」は特性が真であることを証明されたことを示し、「－」は帰納証明が不確定に終わったことを示す。ＢＭＣエンジンが真であることを証明できたのは、３３特性のうち２３特性であった。第５欄は、帰納証明が実施された最大深度を示す。成功裡に終わった証明の深度は、すべて２未満であった。一方、不確定性の証明は、多数の特性において深度２５まで成功しなかった。最後に、第６欄と第７欄は、証明の実行で必要とされた

時間（単位：　秒）とメモリ量（単位：　メガバイト）を示す。時間とメモリの所要量は、深度２５の場合でも非常に少ないこと注目されたい。

【０１０７】

次の項では、効果的な到達可能性の不変式を発見するために行った試みと、帰納証明でこれらの不変式を使用した結果について解説する。

【０１０８】

８．　ＢＭＣとＢＤＤの結合

この項では、ＢＤＤとＳＡＴベースのＢＭＣを結合するための新規なフレームワークについて説明する。その基本概念はいたって大まかで、「ＢＤＤとして計算された状態集合に関する外部情報を利用してＢＭＣの探索を制約する」というものである。ＢＤＤの操作は（時間とメモリの両方において）資源集約的な傾向があり、これが実用使用では問題となる。そのため、ＢＤＤは回路またはＣＮＦの形式に変換される。これにより、ＢＭＣエンジンはＢＤＤを回路／ＣＮＦ制約として処理できるようになる。

【０１０９】

このフレームワークを使用して帰納証明を行う方法は、以下のとおりである。最初のタスクは、到達可能状態の集合の過剰近似を得ることである。これは様々な方法で行うことができる。例えば、設計の「存在」抽象化は、一部のラッチを疑似１次入力として抽出して取り除く方法で行うことができる。こうした抽象化の有力な候補となるのは、順次コアに寄与しない周辺のラッチや、特性信号の依存性閉包の深部に存在するラッチなどである。本質的には、抽象設計は、実設計に存在するパスのスーパーセットを含まなければならない。この要件は、ＢＤＤベースの記号的走査を使用して、到達可能状態の過剰近似集合を得ることによって満たすことができる。また、抽象設計に細密走査を行う代わりに、近似走査手法を使用することもできる（非特許文献２８、２９参照）。これに加えて、ＢＤＤ用の過剰近似手法を使用して、最終ＢＤＤのサイズをさらに縮小することも可能である（非特許文献３０参照）。

【０１１０】

ＢＤＤ（または、一連のＢＤＤ）として到達可能状態の過剰近似集合が得られたら、それをＢＭＣエンジンに適した形式に変換することが次のタスクとなる。このとき、ＢＤＤは回路／ＣＮＦ形式に変換される。この形式の各内部ＢＤＤノードは、ＢＤＤ変数（この場合は、「状態」変数）によって制御されるマルチププレクサーとみなされる。この派生された回路／ＣＮＦを到達可能性制約としてＢＭＣエンジンに追加し、本発明で解説した疑似コード形式の帰納証明プロシージャーとして使用する。

【０１１１】

実用使用では、ＢＤＤベースの到達可能性制約の使用によって帰納証明の完了が可能になり、一方、この制約を使わない証明は不成功に終わったという事例が報告されている。表６（図１１）に実験結果を示す。この実験も、４４０　ＭＨｚ、１　ＧＢのＳｕｎ　ＵｌｔｒａＳｐａｒｃワークステーション上で実施した。この表で、第３～６欄は抽象設計に対するＢＤＤベースの到達可能性分析の結果を示し、第７～１１欄はフル設計に対するＢＭＣベースの帰納証明の結果を示す。実験に使用した抽象設計は、制約なし設計から、特性信号の依存性閉包の深部に存在するラッチを抽象化して除去することによって自動的に取得した。どのケースでも、これに要した時間は１分間未満であった。第３欄は、抽象設計内に存在するフリップフロップ数（＃ＦＦ）とゲート数（＃Ｇ）を示す。これらの数値は、フル設計の対応する数値（第７欄）に比較してはるかに小さいことに注目されたい。第４欄は、回路形式への変換を含む、抽象設計の記号的走査に要した時間（単位：　秒）を示す。第５欄には記号的走査に要した反復回数、第６欄にはＢＤＤの最終的なサイズ（一部のケースでは、さらに過剰近似を行った後のサイズ）を示す。ＢＭＣ帰納証明については、第８欄と第９欄に帰納の検証状況と深度をそれぞれ示す。いずれのケースにおいても証明はきわめて容易に完了できたことに留意されたい。最後に、第１０欄と第１１欄に、ＢＭＣエンジンが費やした時間とメモリ量を示す。

【０１１２】

これらの結果のうち特に注目されるのは、必要とされるＢＤＤの時間とサイズをできるだけ抑えるため、ＢＤＤベース分析では大まかな近似しか行わなかったにもかかわらず、得られた到達可能性制約はＢＭＣの帰納証明を完遂するのに十分な強度を持ち得ていたことである。ＢＤＤ、ＢＭＣエンジンのいずれもこれらの安全特性を個別では証明できないが、本発明のフレームワークと組み合わせることにより、証明が２０秒未満で完了するという驚くべき成果が得られた。

【０１１３】

組み合わせフレームワークを使用した主な動機は、ＳＡＴベースＢＭＣエンジンの本来の機能はパス・ベースで、一方、ＢＤＤベース記号的分析は集合ベースであるという事実に発する。これにより、純粋なＢＭＣエンジンの機能を可能な限りＢＤＤで補完するという利用法が可能になる。このフレームワークはさらに、任意ＣＴＬ特性の反証／証拠を検出する用途にも利用できる。与えられた設計から得た抽象設計にＢＤＤベースの近似モデル検査を実行することにより、「証拠グラフ」のＢＤＤ表現を得ることができる（非特許文献１８参照）。これらの状態は、ＢＭＣエンジンの反証探索に制約を付加するか、優先順位を付けるために使用できる。

【０１１４】

前述の明細書の説明から、当該技術に精通した当業者には本発明には他の変更および異型が可能なことが明らかであろう。したがって、本書では本発明のごく一部の実施例のみが特に説明されているが、本発明の精神および範囲から逸脱することなく無数の変更が可能であることは明らかである。

【０１１５】

【発明の効果】

以上説明したように本発明の有界モデル検査方法によれば、アプリケーションのメモリ所要量がほぼ半減されるため、より大規模な回路にスケーリングすることが可能になり、ＢＭＣの場合はスケーリングできる時間フレーム数も増大する。

【図面の簡単な説明】

【図１】ＢＭＣシステムに含まれる機能を図示する検証プラットフォームの概略図を示す図である。

【図２】（ａ）ループなしの有界パスのケース分割を示す。（ｂ）　（ｋ－ｌ）ループにおける有界パスのケース分割を示す図である。

【図３】回路ベースのＢＣＰプロシージャーの実装に関する疑似コードを示す図である。

【図４】高速含意伝搬のための２入力ＡＮＤルックアップ・テーブルを示す図である。

【図５】１００万含意当たりのブール制約伝搬　（ＢＣＰ：　Ｂｏｏｌｅａｎ　Ｃｏｎｓｔｒａｉｎｔ　Ｐｒｏｐａｇａｔｉｏｎ）時間を記載した表１を示す図である。

【図６】比較におけるＳＡＴ実行時間を記載した表２を示す。

【図７】ハイブリッド・ソルバーのヒューリスティックによって向上したＳＡＴ実行時間を記載した表３を示す図である。

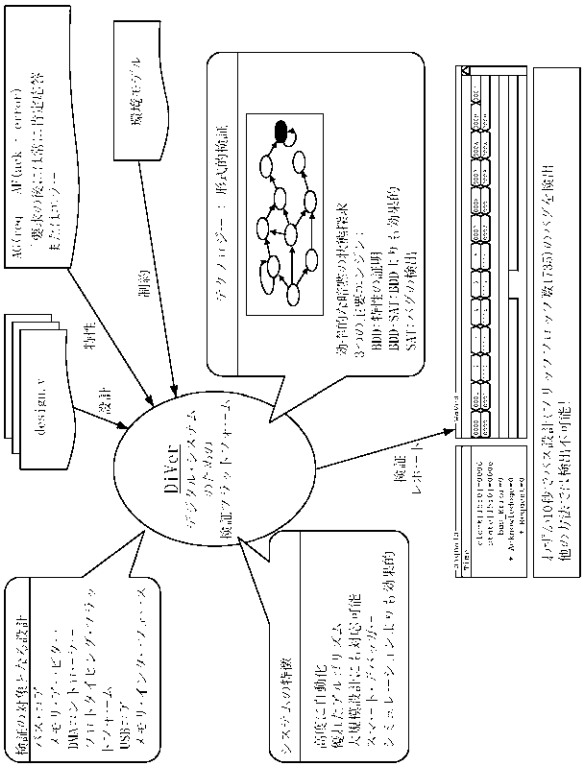
【図８】文節内のリテラル計数の例を示す図である。

【図９】ＢＭＣ検証の結果を記載した表４を示す図である。

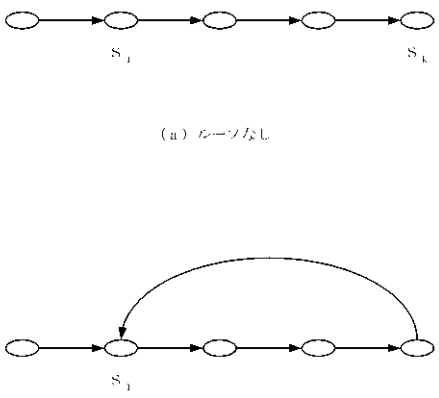
【図１０】到達可能性の制約のない帰納例による証明の結果を記載した表５を示す図である。

【図１１】到達可能性の制約のある帰納例による証明の結果を記載した表６を示す図である。

【図１】



【図２】



【図３】

回路ベースBCPプロシージャー

```
Algorithm imply(vertex, value) {
  assign(vertex, value);
  lvalue = get value(vertex->left);
  rvalue = get value(vertex->right);
  next_state = lookup(value, lvalue, rvalue);
  switch(next state)
   case CONFLICT :
    return 1;
   case CASE SPLIT :
    add vertex(vertex, justification queue);
    return 1;

    ...

   case PROP LEFT AND RIGHT :
    if (imply(vertex->left.next state ->lvalue) &&
       imply(vertex->right.next_state ->rvalue)) {
        return 1;
    }
    return 0;

    ...
  }
 return 1;
```

【図４】

高速含意伝搬のための2 入力ANDルックアップ・テーブル

Current	Next	Action
1, X → X	1, X → X	STOP
0, X → 1	0, X → 1	CONFLICT
X, X → 0	X, X → 0	CASE SPLIT
0, X → X	0, X → 0	PROP FORWARD
X, X → 1	1, 1 → 1	PROP LEFT RIGHT
・・・	・・・	・・・

【図５】

論理式	1次入力	ゲート	BCP時間(秒/100万含意)					
			ハイブリッド	Chaff	CH	構造	CS	HS
230U	5016	228346	1.154	1.589	1.37	0.9	1.76	1.28
252U	5244	242170	1.168	1.586	1.35	0.914	1.73	1.28
272U	5244	238755	1.101	1.411	1.28	0.891	1.578	1.23
274U	3863	179217	1.11	1.52	1.369	0.908	1.67	1.22
275U	5472	255979	1.162	1.566	1.347	0.955	1.64	1.22
276U	5472	256000	1.172	1.571	1.34	0.91	1.72	1.29
405U	6612	325069	1.236	1.605	1.29	0.92	1.74	1.34
406U	6612	325090	1.207	1.639	1.35	0.94	1.74	1.28
407U	6612	325090	1.232	1.694	1.375	0.942	1.798	1.31
409U	6612	325090	1.245	1.639	1.316	0.923	1.775	1.35
435U	6840	338899	1.233	1.668	1.35	0.995	1.676	1.24
2dlx_100	557	33848	1.82	2.22	1.21	0.931	2.38	1.96

表1：100万含意当たりのBCP時間

【図６】

表2：総実行時間におけるハイブリッドソルバーとChaffの比較

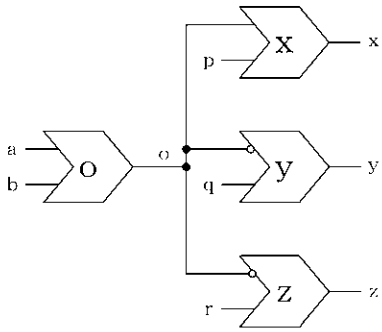
サンプル	1次入力	ゲート	Chaff	H	H_jft	Chaff/H	Chaff/H_jft
充足不能インスタンス							
553I	6384	516779	1893.13	1629.65	639.20	1.16	2.96
35I	6612	571185	1585.08	1102.80	715.20	1.13	2.22
51I	6156	521116	1283.82	1010.90	508.80	1.27	2.52
17I	5700	177037	1190.05	863.60	391.87	1.38	3.04
19I	5928	199710	1072.61	738.30	606.76	1.45	1.77
135I	6840	338899	922.61	326.50	317.00	2.83	2.91
3pipe	198	10816	920.50	245.60	566.90	3.75	1.62
15I	5472	152641	888.86	415.27	443.10	2.14	2.00
131I	6840	338899	880.84	532.10	318.00	1.66	2.77
107I	6612	325090	736.19	367.20	588.00	2.01	1.25
138I	6840	338899	647.71	427.20	283.00	1.52	2.29
137I	6840	338899	621.17	281.10	419.00	2.21	1.48
139I	6840	338899	599.11	441.80	453.00	1.36	1.32
136I	6840	338899	559.08	369.60	384.00	1.51	1.46
105I	1017	91528	517.68	218.90	506.30	2.20	1.08
109I	6612	325090	396.21	437.20	329.00	1.11	1.51
106I	6612	325090	386.86	353.90	317.00	1.08	1.54
105I	6612	325089	471.50	585.00	445.00	0.81	1.06
13I	5214	129978	471.13	353.87	418.25	1.33	2.60
109I	1055	101180	441.23	173.76	229.00	2.54	1.93
108I	6612	325090	353.21	358.16	317.00	1.21	1.25
11I	5016	105572	364.03	230.27	233.80	1.58	1.56
39I	1788	382969	307.59	208.87	120.20	1.47	2.56
101I	979	88088	230.97	141.00	117.00	1.64	1.97
276I	5472	256060	193.21	132.30	104.00	1.46	1.86
37I	1560	358565	171.17	116.99	139.12	1.49	1.23
252I	5244	242170	165.16	148.30	93.50	1.10	1.77
35I	1332	335840	165.20	128.90	76.02	1.28	2.17
275I	5472	255979	162.97	132.10	129.00	1.23	1.26
230I	5016	228346	132.51	100.60	77.00	1.32	1.72
33I	1104	311131	98.61	80.10	42.10	1.23	2.34
31I	3876	288771	92.22	51.30	29.30	1.80	3.15
2dlx_cc_mc_ex_bp_f_new	111	25075	89.07	49.30	72.10	1.81	1.23
272I	5244	238755	47.51	27.76	8.00	1.72	5.94
充足可能インスタンス							
109S	2374	111590	811.12	330.4	182.5	2.45	4.44
105S	2286	131651	432.22	196.4	72.9	2.20	5.93
108S	2367	142168	423.63	766.8	138.5	0.55	3.06
101S	2198	125070	388.35	399.38	111.9	0.97	3.47
102S	2235	127558	341.34	312.7	141.4	1.10	2.38
103S	2242	129818	343.95	481.8	86.7	0.71	3.97
107S	2338	139578	286.1	256.4	178.2	1.12	1.61
104S	2279	132350	270.01	381.4	49.1	0.70	5.50
106S	2323	137230	226.57	198.58	94.1	1.14	2.40
100S	2190	120365	153.11	371.8	97.8	0.41	1.57
2dlx_100	557	33848	120.12	76.8	17	1.56	7.07

【図７】

表3：ハイブリッドソルバーのヒューリスティックによって向上したSAT実行時間

サンプル	1次入力	ゲート		H1	比	E2	H2	H2_Fc	H2_Fc_Fs	H	Chaff/H
充足不能											
553I	638	53677	1893.1	1629.0	1.16	639.	1501.	522.	81	522.	3.6
35I	661	57118	1585.0	1102.	1.13	715.	1127.	1279.	1157.	715.	2.2
51I	615	52111	1283.8	1010.	1.27	508.	1115	553.0	1583.	508.	2.5
17I	570	17704	1190.0	863.	1.38	391.8	621.	370.	907.2	370.	3.2
19I	592	19971	1072.6	738.	1.15	606.	817.	515.	1093.	515.	1.9
135I	681	33889	922.6	326.	2.83	31	51	19	36	31	2.9
3pipe	19	1081	920.	215.	3.75	566.	81.	25	252.	81.	11.2
15I	547	15261	888.8	115.2	2.11	143.	387.	193.	311.	193.	1.5
131I	681	33889	880.8	532.	1.66	31	17	39	35	31	2.7
107	661	32509	736.1	367.	2.01	78	30	41	33	30	2.5
138I	681	33889	647.7	127.	1.52	28	37	53	37	28	2.2
137I	681	33889	621.1	281.	2.21	11	32	50	20	281.	2.2
139I	681	33889	599.1	111.	1.36	15	29	35	27	29	2.0
136I	681	33889	559.0	369.	1.51	38	15	10	50	369.	1.5
104I	101	9152	517.6	218.	2.20	506.	758.	505.	51	188.	2.9
109	661	32509	396.3	137.	1.11	32	28	13	50	28	1.7
106	661	32509	386.8	353.	1.38	31	36	33	26	26	1.8
105	661	32506	471.	58	0.81	11	39	35	17	31	1.2
13I	521	12997	471.1	353.8	1.33	118.2	371.	290.	279.6	181.2	2.6
109I	105	10118	411.3	173.	2.51	22	107	137	467.	173.	2.5
108	661	32509	353.2	358.	1.21	31	22	31	29	22	1.9
11I	501	10557	361.0	230.3	1.58	223.	261.8	153.	302.3	153.	2.3
39I	178	38290	307.5	208.8	1.17	120.	191.0	187.3	115.	120.	2.5
101I	97	8808	230.9	11	1.61	11	198.	39	360.	11	1.9
276	547	25600	193.2	132.	1.16	10	14	12	11	10	1.8
37I	156	35850	171.1	116.9	1.19	139.1	183.	95.	192.	95.	1.8
252	521	24217	165.1	118.	1.10	93.	95	71	77	71	2.3
35I	133	33584	165.	128.	1.28	76.0	119.0	118.	307.	76.0	2.1
275	547	25597	162.9	132.	1.28	12	93	12	10	93	1.7
230	501	22834	132.5	100.	1.32	77	67	83	56	67	1.9
33I	110	31113	98.6	80.	1.23	22.	83.	61.6	65.	12.	2.3
31I	387	28877	92.2	51.	1.80	29.	50.	21.0	58.	21.0	3.8
2dlx_cc_mc_ex_bp_f	11	2507	89.0	19.	1.81	72.	88.	218.	39	19.	1.8
272	521	23875	17.5	27.	1.72	8	20	15	16	8	5.9
充足可能											
109S	237	11159	811.1	330.	2.15	182.	294.	13	199.	13	6.1
105S	228	13165	432.2	196.	2.20	72.	231.	157.	130.	72.	5.9
108S	236	14219	423.6	766.	0.55	138.	264.	183.	228.	138.	3.0
101S	219	12507	388.3	399.3	0.97	111.	176.	105.	129.	105.	3.6
102S	223	12755	351.3	312.	1.10	111.	156.	123.	159.	122.	2.7
103S	224	12981	343.9	181.	0.71	86.	151.	93.	132.	86.	3.9
107S	233	13957	286.	256.	1.12	178.	162.	156.	187.	156.	1.8
104S	227	13235	270.0	381.	0.70	19.	32	138.3	158.	19.	5.5
106S	232	13723	226.5	198.5	1.11	91.	215.	135.0	155.	91.	2.4
100S	219	12030	153.1	371.	0.41	97.	110.	157	107.	97.	1.5
2dlx_10	55	3381	120.1	76.	1.56	17	17	1	11	1	20.0

【図８】



(a)

(a' + o) (b' + o) (a + b + o')
(o' + x) (p' + x) (o + p + x')
(o + y) (q' + y) (o' + q + y')
(o + z) (r' + z) (o' + r + z')

(b)

【図９】

表４：BMC検証の結果

設計 特性	フリップフロップ数	ゲート数	オプション	境界k	証拠	Time(秒)(漸増)	Time(秒)(漸減)	メモリ(MB)
Bus1 P1	1735	9567	-p1 -T -C	18	NO	6620	6789	136
			-p2 -T -C	20	YES	91	99	28
			-p3 -T -C	20	YES	291	326	15.5
			-p4 -T -C	20	YES	11	15	10.5
			-p5 -T -C -H	20	YES	7	8	7.1
Bus2 P1	1997	11033	-p1 -T -C	18	NO	4721	4754	168
			-p2 -T -C	23	YES	270	379	67
			-p3 -T -C	23	YES	602	733	60.7
			-p4 -T -C	23	YES	70	108	21
			-p5 -T -C -H	23	YES	26	51	15
Bus3 P1	2259	12671	-p1 -T -C	18	NO	6369	6357	205
			-p2 -T -C	29	YES	7036	7502	200
			-p3 -T -C	29	YES	2131	3871	87
			-p4 -T -C	29	YES	693	3053	31.1
			-p5 -T -C -H	29	YES	300	1801	37.5
Bus4 P1	2705	16670	-p1 -T -C	18	NO	9320	9402	255
			-p2 -T -C	18	NO	1611	1657	130
			-p3 -T -C	18	NO	1207	1540	91
			-p4 -T -C	18	NO	170	398	58
			-p5 -T -C -H	18	NO	87	159	31
Dma P1	72	328	-p1 -T -C	72	NO	3880	3971	11
			-p2 -T -C	72	NO	1009	1061	9
			-p3 -T -C	200	NO	2	3	8
			-p4 -T -C	200	NO	1	1	8
			-p5 -T -C -H	200	NO	6	6	8
Dmu P1	231	2495	-p1 -T -C	20	NO	3436	5718	27
			-p2 -T -C	20	NO	8.1	1307	19
			-p3 -T -C	50	NO	97	578	29
			-p4 -T -C	50	NO	37	205	21
			-p5 -T -C -H	50	NO	28	242	19
D1 P3	256	1582	-p1 -T -C	16	YES	996	1010	22
			-p2 -T -C	16	YES	19	55	6
			-p3 -T -C	16	YES	3	3	7
			-p4 -T -C	16	YES	1	1	3
			-p5 -T -C -H	16	YES	1	1	2
D1 P4(S)	256	1582	-p1 -T -C	21	YES	8808	9107	12
			-p2 -T -C	21	YES	8.8	1676	18
			-p3 -T -C	21	YES	7	8	10
			-p4 -T -C	21	YES	3	5	6
			-p5 -T -C -H	21	YES	3	1	5
D1 P5(S)	256	1582	-p1 -T -C	25	NO	8595	8978	12
			-p2 -T -C	25	NO	688	737	17
			-p3 -T -C	65	NO	5319	11758	65
			-p4 -T -C	65	NO	3101	3910	46
			-p5 -T -C -H	65	NO	1388	11865	66
D1 P6(S)	256	1582	-p1 -T -C	17	YES	1741	1809	26
			-p2 -T -C	17	YES	127	161	7
			-p3 -T -C	17	YES	1	5	7
			-p4 -T -C	17	YES	1	1	3
			-p5 -T -C -H	17	YES	1	1	5
D1 P7	256	1582	-p1 -T -C	17	YES	691	725	21
			-p2 -T -C	17	YES	32	37	7
			-p3 -T -C	17	YES	2	2	7
			-p4 -T -C	17	YES	0	0	2.8
			-p5 -T -C -H	17	YES	0	0	1.9
D2 P[illegible]	971	10850	-p1 -T -C	21	YES	663	727	17
			-p2 -T -C	21	YES	8	8	22.5
			-p3 -T -C	21	YES	12	11	11
			-p4 -T -C	21	YES	1	1	22.5
			-p5 -T -C -H	21	YES	3	3	21

【図１０】

表５：到達可能性制約を使用しない帰納証明の結果

設計	特性	フリップフロップ数/ゲート数	状況	深度	時間(秒)	メモリ(MB)
D3	p1	90/715	T	2	0.09	0.76
	p2	317/2345	T	0	0.06	1.01
D4	p1	7/13	T	1	0	0.28
	p2	57/794	-	25	24.26	17.2
	p3	57/794		25	24.27	17.41
	p4	57/794		25	26.79	17.37
	p5	57/794		25	24.91	17.29
	p6	53/821		25	36.39	18.56
	p7	24/85	T	0	0	0.28
	p8	50/818	T	0	0.02	0.42
	p9	6/10	T	0	0	0.27
	p10	19/55	T	0	0	0.28
	p11	20/61	T	0	0	0.28
	p12	35/509	T	0	0.01	0.34
	p13	53/821	T	1	0.04	0.64
	p14	53/821	T	0	0.02	0.42
D5	p1	2198/20221	T	0	0	2.32
	p2	2198/20223	T	1	0.3	4.45
	p3	2198/20224		5	7.73	29.1
	p4	2198/20021	T	0	0	2.32
	p5	2198/20226	T	1	0.29	6.32
	p6	2198/20035	T	0	0.05	3.21
	p7	2198/20265	T	1	0.23	6.32
	p8	2198/20266		5	5.87	28.6
	p9	2198/20296	T	1	0.29	6.33
	p10	2198/23333		5	8.93	36.3
	p11	2198/20225	T	1	0.31	6.32
	p12	2198/20238	T	0	0.04	3.2
	p13	2198/20226	T	1	0.3	6.32
	p14	2198/20248	T	0	0.06	3.21
	p15	2198/20221	T	0	0.01	2.32
	p16	2198/20538		5	14.86	31.5
	p23	2201/20259		5	11.63	30.9

【図１１】

表６：到達可能性制約を使用した帰納証明の結果

設計	特性	BDDベースの到達可能性分析				BMCベースの帰納証明				
		フリップフロップ数/ゲート数	時間(秒)	ステップ	BDD	フリップフロップ数/ゲート数	状況	深度	時間(秒)	メモリ(MB)
D5	p3	11/462	1.6	7	131	2198/14702	T	0	0.07	2.72
	p16	115/1005	15.3	12	677	2265/16079	T	0	0.11	2.84
	p23	63/1001	18.8	18	766	2201/16215	T	0	0.1	2.85

フロントページの続き

(72)発明者　アーティ　グプタ
アメリカ合衆国、ニュージャージー　０８５４０　プリンストン、４　インディペンデンス　ウェイ　エヌ　イー　シー　ユー　エス　エー　インク内
(72)発明者　ジジャン　ヤン
アメリカ合衆国、ニュージャージー　０８５４０　プリンストン、４　インディペンデンス　ウェイ　エヌ　イー　シー　ユー　エス　エー　インク内
(72)発明者　プラナブ　アシャー
アメリカ合衆国、ニュージャージー　０８５４０　プリンストン、４　インディペンデンス　ウェイ　エヌ　イー　シー　ユー　エス　エー　インク内

Ｆターム(参考) 5B046 AA08　BA03　JA01