記号処理による設計検証（2）

7M-6

(株)日立製作所　エネルギー研究所
山田直之　小林康弘

1．緒言

定理証明法により論理設計結果を効率的に検証する為の定理証明プログラムを開発している。本プログラムの特徴は、証明過程に論理設計検証に特有の知識を導入し、有効な手順を優先的に選択する機構を実現している点にある。前報では、決定された証明手順に従って証明を実行するプログラムについて報告した（文献（1））。本報では、証明手順を決定する機構、及び全体処理について示すと共に、簡単な検証例について報告する。

2．プログラム構成

定理証明プログラムは、証明手順を決定するブロックと決定された手順に従って証明を実行するブロックから構成している（文献（1））。証明実行ブロックでは、証明戦略として結合グラフ法を採用しており、その下で導出、及び等号調整を実行する。この結果、証明手順決定ブロックでは具体的な証明手順として結合グラフ中の導出可能リンク、あるいは等号調整可能リンクを指定するだけでよい。

3．証明手順決定機構

従来の定理証明プログラムでは、証明手順を制御するために定理の構文的特徴を利用しているのに対し、本プログラムでは、証明すべき定理や、証明に使用する定理、公理の特徴を利用している。また、これらの知識を効率良く利用し、余分な処理時間を少なくする為に、次の２点を実現している。

（1）知識を大局的な方針に関する知識（以下、ストラテジと呼ぶ）と、その下で具体的な手順を与える知識（以下、タクティクスと呼ぶ）に分け階層的に使用する。

（2）知識を事前に定義された関数を使用したルール形式で表現する。

(1) により、証明手順決定の際に探索すべき範囲を限定できる。また、(2) により、各知識を実行可能なプログラムに変換して使用することができる。本プログラムでは、設計記述言語として状態遷移表現言語であるＲＴＳ(文献(2))を使用しており、各知識もこの言語表現を利用して記述する。図１に、ストラテジを決定する為の知識の例を、図２に、タクティクスを決定する為の知識の例を示す。ストラテジ決定に関する知識の末尾の数字は、その知識の優先度を示し、タクティクス決定に関する知識の末尾の数字は、同一ストラテジ内での優先度を示している。

(Mrule-1 (If (and (*Target_literal $x) (*Exist_spliteral $x '(BC DT) '(ST) $y) (*Sour_part_of $x $w) (*Sour_part_of $y $z)) then (Set_cl $y) (Set_ct $z) (Set_tt $w) (Set_strategy 'Unite-ST)) 6)	証明すべきリテラルTLに対し、ブール条件項、デスティネーション項が等価でソース項が異なるリテラルが存在すれば、 そのリテラル、及びそのリテラルのソース項をそれぞれCL,CTに、またTLのソース項をTTにセットし、ストラテジをUnite-ST にする。
(Mrule-2 (If (and (*Target_literal $x) (*Dest_part_of $x $y) (*Has_P_linkz $y 'Expand $z)) then (Set_link $z) (Set_strategy 'Expand-DT)) 2)	証明すべきリテラルTLに対し、そのデスティネーション項が Expand なるグループの等号規則とP-リンクを持てば、 そのリンクインデックスを LINK にセットし、ストラテジを Expand-DT とする。

図１　ストラテジ決定の為の知識例

(Trule-2 (If (Strategy Unite-ST) then (If (*Has_P_link TT 'Simplify $x) then (Paramodulate 'TT $x) (*Sour_part_of TL $y) (Set_tt $y))) 3)	ストラテジが Unite-ST の時、 証明すべきリテラル中の項 TT が Simplify なるグループ名称を持つ等号規則との間にP-リンクを持てば、 そのリンクインデックスによる等号調整を実施し、その結果得られたリテラルのソース項を TT にセットする。
(Trule-3 (If (Strategy Unite-BC) then (If (*Has_P_linkz CT 'Simplify $x) then (Paramodulate 'CT $x) (*Cond_part_of CL $y) (Set_ct $y))) 3)	ストラテジが Unite-BC の時、 証明の目標としているリテラル中の項 CT が Simplify なるグループ名称を持つ等号規則との間に P-リンクを持てば、 そのリンクインデックスによる等号調整を実施し、その結果得られたリテラルのブール条件項を CT にセットする。

図２　タクティクス決定の為の知識例

Design Verification by Symbolic Manipulation (2)

Naoyuki YAMADA, Yasuhiro KOBAYASHI

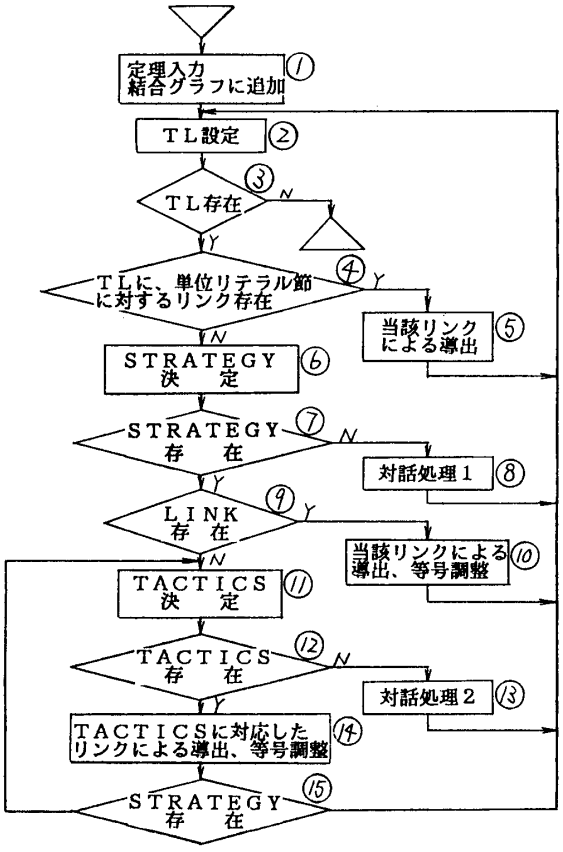
HITACHI, Ltd.

4．全体処理フロー

証明手順決定機構を含む、本プログラムの全体処理フローを図３に示す。図に示すように、証明は、与えられた定理を構成するリテラル毎に実施する。上記した証明手順決定機構は、処理 6、及び 11 で使用される。処理 6 において、実行すべきリンクが得られる場合には、直ちにそのリンクを使用した推論を実施するが、そうでない場合 11 以下の処理により具体的なリンクを決定する。なお、処理 6 、及び 11 において証明手順が決定できない場合は、処理 8 、および 13 において、対話的に証明手順を指定する機構を利用する。

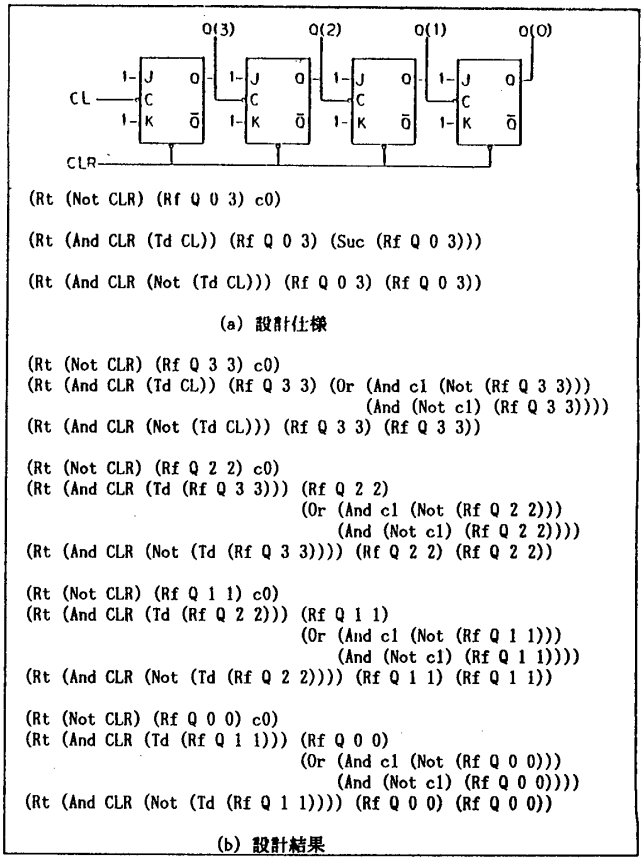
5.検証例

開発した定理証明プログラムを利用してリップルカウンタの設計結果を検証した例について示す。図４にリップルカウンタの回路図、設計仕様、及び、設計結果のＲＴＳ表現を示す。図に示すように、リップルカウンタ自身は比較的簡単な回路であるが、１つのフリップフロップの出力信号が次のフリップフロップのトリガー信号として使用されており、複雑な状態遷移を起こすことから、証明も複雑になる。文献（２）に報告されている、対話型定理証明システムを使用した検証では１０１回の証明手順指示により、この回路を検証している。この中には、表現の簡単化等の処理は数えられていない為、実際には倍以上の証明ステップを要している。本プログラムでは、この証明を２００ステップの推論により自動的に実施することができた。この中、最も複雑な第２項の設計仕様においては、以下の大局的なストラテジを立て、証明することができた。（ａ）デスティネーション項、ソース項のビット連結展開、（ｂ）ビット連結表現を持つリテラルの展開、（ｃ）ブール条件項中の信号の立ち下がり表現の書き換え、（ｄ）ブール条件項の Or 表現への変換，（ｅ）ブール条件項に Or 表現をもつリテラルの展開、（ｆ）ブール条件項の And 要素削減の為の展開。



TL　　：証明すべき定理中のリテラルインデックスを格納
STRATEGY：証明ストラテジーの名称を格納
TACTICS　：証明タクティクスの名称を格納
LINK　　：証明過程でリンクインデックスを格納

図３　定理証明プログラム処理フロー




```
(Rt (Not CLR) (Rf Q 0 3) c0)

(Rt (And CLR (Td CL)) (Rf Q 0 3) (Suc (Rf Q 0 3)))

(Rt (And CLR (Not (Td CL))) (Rf Q 0 3) (Rf Q 0 3))
```

(a) 設計仕様

```
(Rt (Not CLR) (Rf Q 3 3) c0)
(Rt (And CLR (Td CL)) (Rf Q 3 3) (Or (And c1 (Not (Rf Q 3 3)))
                                     (And (Not c1) (Rf Q 3 3))))
(Rt (And CLR (Not (Td CL))) (Rf Q 3 3) (Rf Q 3 3))

(Rt (Not CLR) (Rf Q 2 2) c0)
(Rt (And CLR (Td (Rf Q 3 3))) (Rf Q 2 2)
                              (Or (And c1 (Not (Rf Q 2 2)))
                                  (And (Not c1) (Rf Q 2 2))))
(Rt (And CLR (Not (Td (Rf Q 3 3)))) (Rf Q 2 2) (Rf Q 2 2))

(Rt (Not CLR) (Rf Q 1 1) c0)
(Rt (And CLR (Td (Rf Q 2 2))) (Rf Q 1 1)
                              (Or (And c1 (Not (Rf Q 1 1)))
                                  (And (Not c1) (Rf Q 1 1))))
(Rt (And CLR (Not (Td (Rf Q 2 2)))) (Rf Q 1 1) (Rf Q 1 1))

(Rt (Not CLR) (Rf Q 0 0) c0)
(Rt (And CLR (Td (Rf Q 1 1))) (Rf Q 0 0)
                              (Or (And c1 (Not (Rf Q 0 0)))
                                  (And (Not c1) (Rf Q 0 0))))
(Rt (And CLR (Not (Td (Rf Q 1 1)))) (Rf Q 0 0) (Rf Q 0 0))
```

(b) 設計結果

図４　リップルカウンタ記述例

(1) 山田、木口: 記号処理による設計検証(1)、
第32回情報処理全国大会 4C-1,1986

(2) T.J.Wagner:Hardware Verification:,PhD
Dissertation,CSD Stanford Univ.1977