命題時相論理の決定手続きの比較

Comparison of Propositional Temporal Logic decidion procedure

比嘉 薫[†]　　河野 真治[††]
Kaoru Higa　　Sinji Kono

†琉球大学 理工学研究科情報工学専攻
Faclcuty of Information Engneering, University of the Ryukyus
††琉球大学 情報工学科
Specialty of Information Engineering, University of the Ryukyus

概 要

命題時相論理はハードウェア, ソフトウェアにおいての検証法に有効であることが知られている. 検証の計算量は P-Space であることが知られているが, その詳細は対象となる問題や時相論理の種類, 検証法によって異なる. ここでは Tableau 法と Resultion を取り上げ, 実際にかかる時間について実測し比較と考察を行う.

1 はじめに

近年, 科学技術の発達によりハードウェア ・ソフトウェアは複雑かつ大規模なものとなっている. そのため設計段階において誤りが生じる可能性が高くなってきている. その結果, 設計されたシステムに誤りが無い事を保証するための論理設計や検証手法の確立が重要な課題となっており, 数学的に設計の正しさを保証することができる検証手法が重要視されてきている.

時相論理はその検証手法でも有効な手段の一つである. しかし検証における計算量は P-Space であることも知られている. しかし, その詳細は対象となる問題や時相論理の種類, 検証法によって異なる. 本研究では LTTL(Linear Time Temporal Logic),ITL(Interval Temporal Logic) の 2 つの論理に対して, Semantic Tableaux,Resolution という 2 つの決定手続きを実装し, 充足性のチェックにかかる計算時間を実測, 比較しその違いについて考察する.

本研究で用いる Temporal operator は 5 種類あり LTTL では □(allways), ◇(sometime) ITL では&(chop),|(bar) を用いる. またそれぞれの論理で用いる operator の◯(Next) がある.

2 LTTL

LTTL では時間を直線的で,1 クロックづつに分ける離散的な時間を考える. この論理は終了時間が存在しない無限の時間で考える論理である.

2.1 Semantic Tableaux

定義 1 s を状態として,$v_s(A)$ は命題論理式 A に対応する状態 s での真理値を表すとする.*LTTL* では, 状態は時刻に対応する. 時刻 i の状態を s_i とする.

$\Box A$: 最初の状態 s_0 から引き続く, 任意の状態 s_i において $v_{s_0}(A) = T$ かつ $v_{s_i}(A) = T$ が成り立つときである.

$\Diamond A$: s_0 から引き続くある 状態 $s_i(0 \leq i)$ において $v_{s_i}(A) = T$ が成り立つときである.

$\bigcirc A$: s_i の次の状態 s_{i+1} で $v_{s_{i+1}}(A) = T$ がなりたつときである.

これに従来使われている論理記号,$\vee, \wedge, \neg, \Rightarrow$ を用いる.

まず Semantic Tableaux と呼ばれる決定手続きについて説明する．また, この手続きで用いる temporal operator の展開規則を表 1に示す．

αルールにおける展開で導出された式はAND,βルールはOR で結合しているものとする. 古典論理に関する展開については省略する．

このルールにより, 論理式をツリー状に展開することで充足性のチェックを行う. Semantic Tableaux では展開したツリーの全ての葉に以下のマークを示せば終了である．葉は次のように分けられる．

- モデルが存在することを示す **open**
- モデルが存在しないことを示す **close**
- ある部分ツリーまでのループを示す$*$

また, この論理では考える時間が無限なので, 無限にイベントが延期される場合を検出する eventuality のチェックを行わなければならない.eventuality のチェックとは, 任意のループで, 起こるべき事象が今までに発生しているかどうかを調べる事である. これは展開を行ってできた葉の親をたどっていき, 起こるべき事象が過去に発生していたかどうかを調べることで十分である [1] .

2.2 Resolution

この検証方法は与えられた式を Conjuctive Normal Form(CNF) と呼ばれる形式に変換し,resolution によって CNF を減らしていくことにより, 充足性のチェックを行う [1].

定義 2 命題変数と, その否定 $p, \neg p$ をリテラルという.

next operator のついた式やリテラルを項という.

always operator と *next operator* とリテラルだけでできたものを元素項といい, さらに *sometime operator* がついたものを条件項という.

元素項の集合でできたものを節といい, 項の集合でできたものを類似節という.

αルール	$\alpha 1$	$\alpha 2$
$\Box A$	A	$\bigcirc\Box A$
$\neg\Diamond A$	$\neg A$	$\neg\bigcirc\Diamond A$

βルール	$\beta 1$	$\beta 2$
$\Diamond B$	B	$\bigcirc\Diamond B$
$\neg\Box B$	$\neg B$	$\neg\bigcirc\Box B$

表 1　**LTTL** の展開規則

もとの論理式を節で表したものを *CNF* という.

Resolution の展開規則は, 次のようになる.

全ての論理式は $P_i \wedge \Box Q$ という形の CNF に変換することができる. この CNF を使って Resolution を考える (P と Q は節) [4] .

CNF に変換する過程で P,Q がリテラルでない場合の $\Box(P \vee Q)$ という形のときは $(\Box P \vee \Box Q) \Rightarrow \Box(P \vee Q)$ と右向きにしか論理が成り立たないので,$P \vee\ Q$ を新しい変数 R に置き換える. よって式は $\Box(P \vee Q) = \Box R \wedge (\neg R \vee P) \wedge (\neg R \vee Q)$ という CNF に変換される.

ある節 C', C'' において任意のリテラル l が

$$C' \ni l \text{ かつ } C'' \ni l^c$$

なら Resolution して得られる節 C は

$$C = \{C' - l\} \cup \{C'' - l^c\}$$

となる.

節が $\Box, \Diamond$ の場合は, 節を現在時刻と次の時刻に展開する.

Resolution での終了条件は,CNF S_i を resolution して得られる S_{i+1}が $S_i = S_{i+1}$となるか,$S \ni [](nil)$ となった場合である. もし後者の場合ならこの論理式は充足可能 (unsatisfiable) でない.

3 ITL

ITL では真理値は時間区間に対応して決まる.

定義 3 & を任意な 2 つの時区間の結合とし, 時区間 t での A の評価を $v_t(A)$ とする. 論理式での時区間の結合 *(chop)* も同じ記号& で表す.

$v_t(P\&\ Q) = v_{t'}(P) \wedge\ v_{t''}$ となり, $t = t'\&\ t''$

また,*chop* の双対である *bar* $|$ を次のように定義する. $P \mid Q = \neg(\neg P\&\ \neg Q)$

さらに $\bigcirc$ については A より *1* 時区間短い式で考える.

ITL では tableau method の展開を時間が **0** である **emp** と 0 以外の時間である **more** に分けて考える [2] [3] .

ITL では終了時間が存在するため,eventuality に関しては, 展開終了時刻にその事象が起こっているなら, その論理式での eventuality は満たされていると考えることができるのでチェックを行う必要がない.

3.1 Semantic Tableaux

次に $\Box p, \Diamond p$ ITL への変換を示す.
$\Diamond p = true\&\ p$
$\Box p = (false \mid p)$
この変換を用いて ITL の展開を行う.

ηルールにおける展開で導出された式は $P\&Q$ で考えると事象 P が発生した後に事象 Q が発生しなけらばならないので AND 連結,γルールはその双対なので OR 連結となる.

ITL もまた LTTL と同様にツリーを構築し, 終了条件も LTTL と同じものになる.

3.2 Resolution

ITL での resolution を考える. これも CNF への変換を考える. まず, ここでは $\Box, \Diamond$ を chop,bar へ変換する. chop$P\&Q$ の場合を考える. この式を emp,more に分けて考えると emp の場合, 時刻が 0 なので $P \wedge Q$ となり,more の場合 P が発生していると考えるので,$more(P)\&Q = more(P\&Q)$ となり,Q が起こるのが未来に延長されていると考えることができる.

次に bar$P \mid Q$ の場合は emp(P) のとき, 事象 P または Q が発生していればいいので $P \vee Q$ となり,more(P) のときは事象 P が発生しているか,Q が発生しているかを考えればよいので $more(P) \mid Q = more(P \mid Q)$ と変換することができる.

よって式は次のように変換して resolutoin を行う.
$P\&Q = (emp(P) \wedge Q) \vee (more(P\&Q))$
$P \mid Q = (emp(P) \vee Q) \wedge (more(P \mid Q))$

これを allways,sometime operator に適用し変換すると次のようになる.

ηルール	$\eta 1$	$\eta\ 2$
$P\&Q$	$emp(P)\&Q$	$more(P)\&Q$
$\neg(P \mid Q)$	$emp(\neg P)\&\neg Q$	$more(\neg P)\&\neg Q$

γルール	$\gamma 1$	$\gamma 2$
$P \mid Q$	$emp(P)\vert Q$	$more(P)\vert Q$
$\neg(P\&Q)$	$emp(\neg P)\vert\neg Q$	$more(\neg P)\vert\neg Q$

表 2　ITL の展開規則

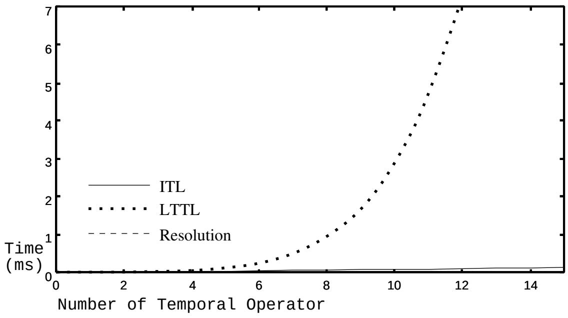

図 1　Allways operator の個数

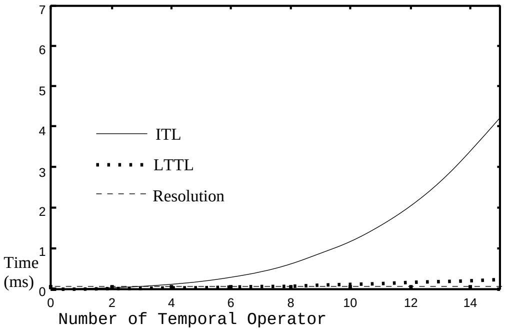

図 2　Sometime operator の個数

$\Diamond P = ((true \vee more(true\&P) \wedge P)$
$\Box P = ((false \vee P) \wedge (more(false|P)))$
となる.

Resolution 手続きや終了条件は,LTTL の場合と同様である.

4　実装評価

これらの論理を実装し, 充足性のチェックにかかる時間を計りその結果をグラフに示した. 図 1 , 図 2 は単純な allways operator,Sometime operator のネストにかかった時間をそれぞれに示す. またいろいろな論理式を実際に展開し, それにかかった平均時間を表 3に示す. かかった時間にある **** は計測した CPU のメモリ不足のため計測できなかったことを示している.

LTTL の resolution に関しては,eventuality のチェックが完全でないため, sometime operator の入った論理式は展開できないものが数多くあった.

図 1, 図 2からわかることは,LTTL と ITL では allways ,sometime operator のネストに関しては対称的な結果となった. それに対し,resolution での展開は temporal operator のネストは CNF に変

LTTL	ITL	LTTL の Resolution
$(p \Rightarrow \bigcirc(\neg p)) \wedge \Box(\neg p \Rightarrow \bigcirc(p \wedge \Box q)) \wedge \Box(p \Rightarrow \bigcirc(\neg p))$		
0.01	0.12	0.05
$(p \Rightarrow \bigcirc(\neg p)) \wedge \Box(\neg p \Rightarrow \Diamond(p \wedge \Box q)) \wedge \Box(p \Rightarrow \Diamond(\neg p))$		
0.065	0.92	2.67
$\Box(a \Leftrightarrow \Diamond a) \Leftrightarrow \Diamond(\Box(a \wedge \Diamond(\neg a)))$		
3.83	0.67	0.07
$\neg(\Box(a \Leftrightarrow \Diamond a) \Leftrightarrow \Diamond(\Box(a \wedge \Diamond(\neg a))))$		
1.57	0.77	0.31
$\Box(\neg\Box a \Leftrightarrow \Diamond a) \Leftrightarrow \Diamond(\Diamond(b \wedge \Diamond(\neg b)) \Rightarrow \Box(\Diamond(\Diamond c \vee \Box(\neg c))))$		
0.48	0.06	****

表 **3**　いろいろな論理式の展開にかかった時間 [s]

換する時にネストの省略を行っている. そのため semantic tableaux よりは高速に展開が行えたと考えられる.

しかし, いろいろな論理式の展開を行った表 3で分かるように operator の組み合わせが増加すると,resolution では展開を行うことができなくなった. それに対し,LTTL,ITL では時間は増加しているが展開はほとんど可能であった.

LTTL,ITL は変数や operator の数が増加すればかかる時間も指数オーダーでほとんど増加している. しかし, 一部の式では高速に行えるものもあった.

Resolution については現在,CNF は線形リストで表している. そのため resolution できないものもリストに入っているので resolution の対象を探すのに, 時間がかかっている. また, 次の時刻に対して展開を行うたびに, リストが大きくなる可能性があり複雑な式ではメモリ不足の為に展開を行うことができなくなったと考えられる.

その解決策としてリストをハッシュ化することで,clashed literal を効率的に探索し,resolution を行う回数を減らす事ができればより高速に実行できると考えられる.

5　結論

本研究では, 各種の時相論理の自動検証を行った. まだどの論理での充足性のチェックに temporal operator が増加すると, 時間がかかってしまう. そのなかで resolution はまだ効率が悪く展開を行えない式がたくさんある. しかし, 与える式によっては semantic tableaux よりも高速に行えるものもあった.

これからの課題として, まず検証系の実装自体の効率が悪いので高速に行えるようにする. さらにハッシュ化, 並列化,BDD 化など考慮し高速化を図る.

参考文献

[1] M.Ben-Ari “Mathematical Logic for Computer Science” pp11-91, pp.64-77, pp200-243, 1992.

[2] Sinji Kono, “A Combination of Clasual and Non Clausal Temporal Logic Program,” July 12 1993.

[3] Sinji Kono, “Automatic Verification on Interval Temporal Logic,” Apr 24 1994.

[4] G.Venkatesh SSE Group “A Decision Method for Temporal Logic Base on Resolution”