ICOT Technical Memorandum: TM-0478

TM-0478

時制論理とペトリネット

木位田真一，内平直志，松本 一教

March, 1988

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

時制論理とペトリネット

本位田 真一，内平 直志，松本 一教

1. はじめに

近年のＡＩ技術に関する関心の高まりに伴い，その応用技術のひとつであるエキスパート・システムが多種多様な分野で構築されるようになってきた．それらの分野の中では，対象とする事象を扱うために時間に関する記述が必要な場合がある．たとえば医療の分野では，患者の病状が時間の経過とともに変わっていき，診断時には，その時間経過による症状の推移が重要な要素となる．したがって，医療診断エキスパート・システムにおいては，時間の表現およびその推論手続きは重要なテーマであり，多くのシステムが構築されてきた．また，電力系統や発電所などの外界を制御する計算機による制御システムにおいても，エキスパート・システムの有効性が認められているが，この場合においても，刻々と変化する外界の状態をいかに表現し効率的に推論していくかが大きな問題として挙げられている．

このように，扱う対象が人間系あるいは機械系であろうと，エキスパート・システムが時系列とともに変化するデータを扱うためには，それらの知識表現およびそれに対する推論手続きの確立が必要となってくる．さらに，時間のもつ性格上，厳密性が要求されるのは当然である．そのために時間に関する表現力を高めるのは比較的容易ではあるが，それに対する推論手続き（この場合には，知識表現が矛盾なく正しいことも検証可能でなくてはならない）が複雑になり，実現が困難になることが課題として認識されている．従来よりこのような課題を克服するために多くの研究が行なわれてきたが，最近，時制論理がこのような厳密な知識表現と正しさの検証を可能とする推論手続きを有しているものとして注目されている．

本稿では，時制論理について，その概要，知識表現手法および推論手続き，さらにペトリネットとの関係について述べる．

ほんいでん　しんいち，うちひら　なおし，まつもと　かずのり　㈱東芝　システム・ソフトウェア技術研究所
〒210 川崎市幸区柳町70

2. 時制論理

時制論理は様相論理のひとつであり，通常の古典論理に時間の概念を導入した論理体系である．時制理論においては，時間によってその論理式の真偽値が変化するが，それを扱うために種々の時制演算子が提案されている．たとえばMannaらは時制演算子として次の

○f （次の状態でfが真である）

◇f （将来fが真である）

f1Uf2（f2が真になるまで常にf1が真である）

□f （常にfが真である）

を命題論理に追加した時制命題論理を提案している[6]．

時制論理には，上のような時制演算子の種類に

よるものも含めて，多くの体系が従来より提案されてきた．それらの体系は大きく

① 線形時間体系

② 分枝時間体系

に分類することができる．

これらの体系においては，体系が異なると同じ時制演算子でも解釈が異なってくる．たとえば◇Pにおいて線形時間体系であると，いつかは不明であるが，将来必ずPが成立するという意味であり，分枝時間体系であると，時系列によってはPが必ずしも成立するとは限らないという表現も可能である．

また，この分類以外にも，過去，未来いずれの時制にもとづいているか，あるいは限定子や自由変数が扱えるかどうかなどによる分類も考えられる．

体系によっては，与えられた論理式が妥当であるかどうか決定する手続きが存在する場合がある．存在する場合，その体系は決定可能であると呼び，知識の検証等を行なう場合，望ましい性質である．

一般に論理体系が強力になるにつれ，決定手続きに要する計算量も増大する．たとえば，ＰＴＬ(Propositional Temporal Logic：時制命題論理）は決定可能であるがＰＴＬに変数を導入すると，決定不能になることが知られている．

また，表現力としては同等であっても，決定手続きの計算量が異なる場合も知られており，論理体系を選択する場合には，注意すべきである．

3. 時制論理による知識表現

通常の一階述語論理にもとづく体系を用いて，知識の表現を行なう場合，時間の概念が陽に含まれていないため種々の困難が生じることがある．

たとえば，変数Iの値を2増やすといった表現は，通常の手続的プログラム言語では$I=I+2$のように記述されることが多いが，等式と見れば論理的に矛盾したものである．同様に，変数Iと変数Jの値を交換するといった表現は，$(I=J)\wedge(J=I)$という形式では表現することはできない．

その他にも，データベースの記述等においては通常の記述では，時間の経過に伴う状況の変化に対応することができず矛盾をきたすことがある．これまでの方法では，新たな事実の追加によりデータベース全体に矛盾をきたすような場合は，過去の事実の削除ということにより，全体の無矛盾さを保っていたが，このような方法では過去の事実をまったく無視してしまうことになり，十分な推論が行なえないことがある．

ＰＴＬを用いて，最初の例を表現してみると，$\bigcirc I=I+2$(現在のIの値を2増やしたものが次の時間でのIの値である)，$(\bigcirc I=J)\wedge(\bigcirc J=I)$（現在の$J$の値が次の時間の$I$の値，現在の$I$の値が次の$J$の値）のように容易に表現することが可能である．

ＰＴＬの採用により，時間に関する表現が容易に行なわれるようになったが，“偶数番目でＰが成立する”等の表現はＰＴＬでは記述することができず，さらに表現力のある体系が求められる．拡張の方法としては，時制述語論理の採用，すなわち変数の導入を行なうことも考えられるが，これは不完全な体系（妥当な論理式であっても証明が存在しない場合がある）になることが証明されており，知識の検証を行なう場合には問題が生じる．そこで Wolper は，ＰＴＬと正規文法を融合させた完全な体系 ETL(Extended Temporal Logic)を考案し，並列プログラムの同期部分の仕様を表現することに使用した[11]．これは，先にＰＴＬでは表現不可能な例として挙げたものが，正規表現では，(P；True)* のように容易に表現できることに注目したものである．

区間時制論理（ITL：Interval Temporal Logic）も，ＰＴＬの拡張として考案されたものである．これは，時間に関する記述を各時点を単位とする方法から，あるまとまり（区間）としてとらえる立場へと変化させたものである．すなわ

ち，ＰＴＬでは各時点λ_iにおいて命題記号の真偽値を考えたが，ＩＴＬでは時区間$\lambda_0 \cdots \lambda_i$での命題の真偽値を問題にする．ここでは Moszkowski の提案したＩＴＬについて紹介する［７］．このＩＴＬでは，通常の命題論理の記号以外に，○（次に），；（ひき続き）という時制記号をもつ．解釈は，区間に対してなされ，○Ｐは，次の区間$\lambda_i \cdots \lambda_i$においてＰが真となるとき真となる．Ｐ；Ｑは，ある$j$が存在して，$\lambda_0 \cdots \lambda_j$でＰが真となり，$\lambda_j \cdots \lambda_i$でＱが真になるとき真になる．Moszkowski は，上で定めた記号を用いて種々の有効な記号を定義し，論理回路の表現に適した体系にまとめあげている．

たとえば，empty≡～○true,

skip≡○empty

と定義すれば，前者は長さ０の区間λ_0においてのみ真となり，後者は長さ１の区間$\lambda_0 \cdot \lambda_1$においてのみ真となる論理式となる．これを用いて，２単位時間の後に，Ｐが成立する時区間があるということを，skip；skip；Ｐのように表現できる．

ＩＴＬは，ＰＴＬを真に含む体系ではあるが，不完全であることが示されており，完全な体系としては，命題Ｐの真偽値が区間の最初の時間で決定されるという性質（局所性という）を課した体系である局所ＩＴＬが提案されている．

ＩＴＬに類似した体系として，Kowalski によって event calculus の提案がなされている［５］．これは，ある時点でのみ成立するという知識が表現できないという欠点はあるが，非単調論理との融合がなされており，知識表現に用いる体系としては有効性が期待できる．

4. 時制論理の推論手続き

時制論理における推論手続きとは基本的には与えられた論理式の真偽値を判定する定理証明系を意味する．本節では，具体的にどのように定理証明系が動作するかについて述べる．時制論理における定理証明系は数多く提案されているが，その中でタブロー法［11］と non clausal resolution 法［１］を紹介する．

タブロー法とは与えられた論理式を満たす状態遷移図を作成する手続きであり，もし空の状態遷移図を作成した場合には，その論理式が充足不可能であるとみなすものである．以下に手順を示す．

まず，与えられた論理式Ｆが妥当であることを証明するために，¬Ｆが充足不可能（いかなる解釈によっても¬Ｆが真にならない）ことを証明する．その時，¬ＦをＦ′としてＦ′を現在を示す論理式Ａと次の時点以降を示す論理式Ｂに次のように分割する．

$F' \rightarrow A \wedge \bigcirc B$

この分割をグラフに対応させる．さらにＢに対しても同様の分割を行ない次々にグラフを構築していく．これをモデルグラフと呼ぶ．次にそのグラフを吟味して必要に応じてノードやアークを削除し，最終的にグラフが空になればＦ′が充足不可能であることが証明できたことになる．逆に与えられた論理式が充足可能ならばグラフが残る．このグラフは状態遷移図であり，プログラムの実行順序を表現しているとみなすことができる．したがって，このタブロー法自体をプログラム合成手法とみなすことも可能である．

タブロー法は時制命題論理に対する証明手続きとしては一般的であるが，その効率や述語論理への拡張が困難であるなどの欠点がある．これらの欠点を解消する手法として Abadi と Manna は non clausal resolution を提案している．non clausal resolution は，通常の resolution のように標準型に変換することなく，論理式に直接 rule を適用する手法である．この手法の利点として，証明が短くて済むこと，証明が人間に対してより解りやすいということが挙げられる．この方法では通常の resolution と同様に論理式Ｆの証明を refutation procedure として行なう．すなわち，論理式Ｆの妥当性を証明する場合，$S_0 = \sim F$より出発し，次々に rule を適用することにより，

$S_0 \cdots S_n$=false なる有限な列が構成できれば，Fは妥当な論理式である．ruleはsimplification ruleとresolution ruleの2つから構成される．

まず，simplification ruleであるが，これは論理式の簡約化に相当し，～(否定)を最も内側に移動する(例～$(u \wedge v) \rightarrow \sim u \vee \sim v$，～$\Box u = \Diamond \sim u$等)．通常のresolution ruleは

R[A<u>，B<u>]→A<true>∨B<false>

なる形で与えられ，これは論理式A，Bがともにuという出現を含む場合，そのあるひとつの出現をそれぞれtrue，falseで書き換え，それらをorで結んだものをresolventにするという操作である．Abadiらは，時制演算子を考慮したresolution ruleを示している．この手法の健全性および完全性もAbadiらにより証明されている．また，通常のresolutionの効率化のために考案された手法を応用することにより，本手法の効率化を行なうことも期待できる．

5. 時制論理とペトリネット

本章では時制論理，特に線形時間にもとづく時制命題論理とペトリネットの関係を論じる．

5.1 システムの構造と制約

世の中の多くのシステムは並列性をもっている．すなわち，システムを構成する個々の要素は，互いに同期をとりながら各々自律的に動く．これを並列(同時進行)システムと呼ぶ．特に現在は，コンピュータネットワークが発達し，複数の計算機にまたがる分散処理システムの需要はますます高まりつつある．

このような並列システムの構造の表現（すなわちモデル化）に適しているのがペトリネットである[8]．ペトリネットの利用形態としては，次の2つが考えられる．

①モデル化に利用する．

②モデル化したペトリネットを解析してシステムの性質を調べる．

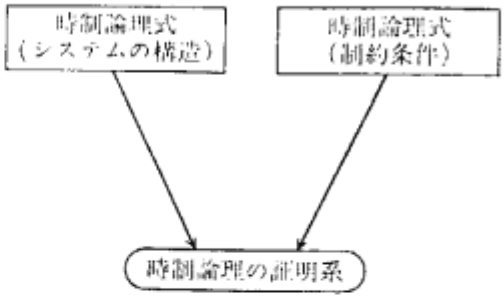

図1 時制論理＋時制論理

ただし，ペトリネットは記述能力は高いが，解析能力はあまり高くない．通常，ペトリネットはモデル化のための表現形式としてのみ使う場合が多い．

一方，時制論理式はシステムの構造とは無関係にシステムの満たすべき性質を記述することができる．そこで時制論理式で並列システムに対する制約条件を与え得る．たとえば，プロセスPがデッドロックに陥らない条件として□◇P（Pは無限回起こる）のように書ける．

このとき，並列システムの構造も時制論理式で記述できれば，そのシステムが時制論理式で与えられた制約条件を満たし得るかどうかを形式的に証明することができる(図1)．(セマフォやモニタの構造を時制論理で記述し，システムのfailure free性を検証したものに[2]がある)

ところが，時制論理は制約条件の記述には適しているが，システムの構造自体の記述に対しては必ずしも適しているとはいえない．ペトリネットの方が，はるかに直観的でわかりやすい場合が多い．

たとえば，図2のような簡単なペトリネットで表現できる同期式通信モデルを時制論理式で記述すると，図3のようになる．

ペトリネットのグラフ表現には，2つの型のノードがある．丸“○”はプレース，棒“|”はトランジションと呼ぶ．プレースの中の黒丸は，トークンである．トランジションへ入力している矢印の元にあるプレースのすべてにトークンが存在すれば，トランジションは発火可能となる．トラ

ンジションが発火すると，入力側の各プレースのトークン1つが出力側の各プレースに移動する．

ここで，時制論理式が繁雑と思われる部分は＊の下線部分である．すなわち，ペトリネットでは発火するトランジションに無関係なプレースのトークンの状態は変わらないわけだが，時制論理では各時間毎に真偽の解釈が異なるので，状態が変わらないということも明確に記述しなければならない．複雑な並列システムでも，ある時点で変化する部分はごく一部であり，ほとんどの部分は変化しない．ところが，時制論理ではその変化のない部分も1つ1つ記述する必要がある．（これは一種のフレーム問題である．この問題を解決するために，時制論理に非単調性を導入した論理も提案されている．[9]）

このように，システムの構造の記述は，時制論理よりペトリネットの方が簡潔で明解である．そこで，システムの構造はペトリネット，制約条件は時制論理で与えるのが良い方法である(図4)．

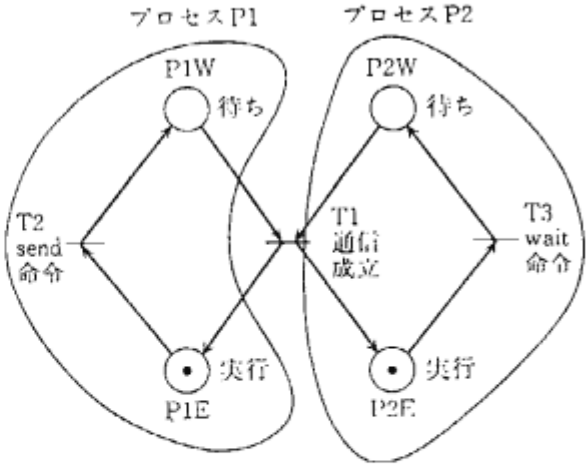

図 2　同期式通信モデル

ペトリネットと時制論理を結合するポイントは，時制論理における原始論理式をペトリネットのどこに対応させるかにある．ここでは，2つの方式を考察する．

5.2　原始論理式＝プレースのトークンの存在

時制命題論理に限定した場合，片井・岩井ら[3][4]は，プレースPにトークンが存在することを命題Pの意味とした．たとえば，図5のペトリネットに対する各命題の解釈は，

P3，P4，P5は真

P1，P2，P6は偽

である．

ここでは，1つのプレースに複数のトークンが存在することのない安全な（safe）ペトリネットを対象とする．また1回に発火できるトランジションは1つである．

図5のペトリネットにおいて，P1にトークンが存在したならば，次の時点ではP2にトークンが存在しなければ

（命　題）

P1E＝プロセスP1が実行状態である

P1W＝プロセスP1が待ち状態である

P2E＝プロセスP2が実行状態である

P2W＝プロセスP2が待ち状態である

T1＝通信成立

T2＝プロセスP1がsend命令を実行

T3＝プロセスP2がwait命令を実行

$$\Box((P1E \wedge T2) \supset \bigcirc P1W) \vee ((P1E \wedge \underset{*}{\sim T2}) \supset \bigcirc P1E))$$

$$\wedge\Box((P2E \wedge T3) \supset \bigcirc P2W) \vee ((P2E \wedge \underset{*}{\sim T3}) \supset \bigcirc P2E))$$

$$\wedge\Box((P1W \wedge P2W \wedge T1) \supset \bigcirc(P1E \wedge P2E)$$

$$\wedge((P1W \wedge \underset{*}{\sim T}) \supset \bigcirc P1W)$$

$$\wedge((P2W \wedge \underset{*}{\sim T1}) \supset \bigcirc P2W))$$

$$\wedge\Box(\sim(P1E \wedge P1W) \wedge (P1E \vee P1W))$$

$$\wedge\Box(\sim(P2E \wedge P2W) \wedge (P2E \vee P2W))$$

図 3　同期式通信モデルの時制論理表現

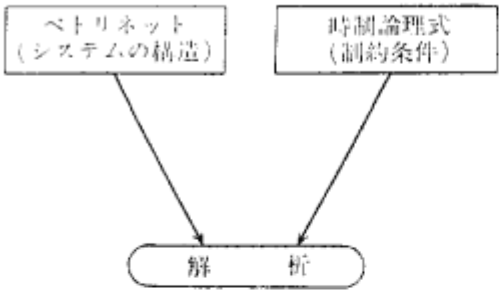

図 4　ペトリネット＋時制論理

ならないという制約条件は

□（P1⊃○P2）　　(1)

のように書ける．

片井・岩井らは，この定式化にもとづいた並列システムの検証体系［4］を示した．また，ペトリネットの可達木と，タブロー法で生成されるモデルグラフを対応づけることにより，デッドロック状態に陥らないためのトランジションの発火禁止則を導く手法［3］を提案した．

5.3　原始論理式＝トランジションの発火

ペトリネットでは，入力プレースのトークンの存在がトランジションの発火条件である．これ以上のトランジションの発火条件の表現については，ペトリネットは何も提供していない．発火条件の記述力を高めるため，さまざまなペトリネットの拡張が提案されている．ここでは，トランジションTが発火することを原始命題Tと対応づけ，トランジションの発火順序を時制命題論理式で表現することを考える．複数の発火可能なトランジションが存在した場合，その中のどれを選択するかの制約条件を時制命題論理で実現できる．ここでも，1回に発火できるトランジションは1つとする．図5の場合，トークンがP1にあるうちは，T4を発火しないという制約条件は

□（T1⊃～T4UT2）

と書ける．

この定式化により，トランジションの発火順序に関する制約を時制命題論理で表現する方法の応用例としては，並列プログラムのスケジューラ生成への適用［10］がある．また，トランジション

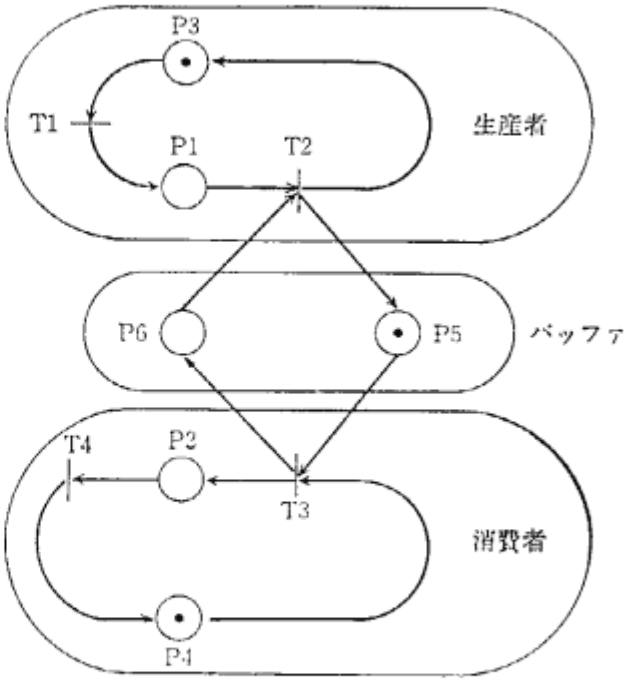

図 5　バッファ1の生産者/消費者問題［3］より引用

をプロダクションルールとみなせば，プロダクション・システムのルールの競合解消戦略の仕様記述にも適用できる．

ただし，この定式化には，5.2のような時制論理とペトリネットを統合した検証体系は提案されていない．

6.　む　す　び

本稿では，厳密な時間に関する知識表現とその推論手続きを備えている時制論理について述べた．推論手続きである定理証明系のもつ大きな課題である計算量の問題に対しては，ハードウエアの進歩は決して根本的な解決策にはならないが，実用性の観点からは大きな変革をもたらしている．たとえば定理証明系でもある Prolog の場合には，数～数十 klips で走行する処理系が一般的であるが，専用ハードウエアによって 500～1000 klipsの性能が得られようとしている．すなわち，従来1時間費やしていた問題に対して 500～1000 倍の高速化によって数秒で解が得られるならば，定理証明系は実用性を帯びてくる．

また，時制論理の応用は単に時間に関する知識

表現にとどまらず，本稿でも一部述べたように，

①プロダクション・ルールに対するメタ知識

②ハードウエア・ロジックの検証

③並列プログラム部品の結合

など，多岐にわたっている．しかし，これらの応用分野において真に実用に供するためには，効率的な推論手続きを構築する技術が必要不可欠である．

参考文献

[1] Abadi, M. and Manna, Z.: A Timely Resolution, Proc. of the Symp. on Logic in Computer Science, Cambridge, MA, 1986

[2] Karp, R. A.: Proving Failure-Free Properties of Concurrent Systems Using Temporal Logic, ACM TOPLAS, Vol. 6, No. 2, 1984, 239-253

[3] 片井　修，岩井壮介：非同期同時進行システムに対する時制論理に基づくスケジューリング則の構成，計測自動制御学会論文集，Vol. 18, No. 12, 1982, 56-63

[4] 片井　修，川井宏弥，多久島郎，岩井壮介：並列プロセスの動態に対する時制論理に基づく記述・検証体系，システムと制御，Vol. 25, No. 8, 1981

[5] Kowalski, R.: A Logic-based Calculus of Events. New Generation Computing, 4, 1986, 67-95

[6] Manna, Z. and Pnueli, A.: Verification of Concurrent Programs, Part1: The Temporal Framework, STAN-CS-81-836, Stanford Univ., 1981

[7] Moszkowski, B. C.: Reasoning about Digital Circuits, PhD Thesis, Department of Computer Science, Stanford Univ., 1983

[8] ピータースン(市川，小林共訳)：ペトリネット入門，共立出版，1984

[9] 佐伯元司：非単調命題時間論理とその形式的仕様記述への応用，情報処理学会論文誌，Vol. 28, No. 6, 1987, 547-557

[10] Uchihira, N., Kasuya, T., Matsumoto, K., Honiden, S.: Concurrent Program Synthesis with Reusable Components Using Temporal Logic, Proc. of Logic Programming Conference '87, ICOT, 1987

[11] Wolper, P. L.: Specification and Synthesis of Communicating Processes Using an Extended Temporal Logic, PhD Thesis, Department of Computer Science, Stanford Univ., 1982