ICOT Technical Memorandum: TM-0479

TM-0479

時間論理ベースの形式的仕様記述言語PTS

川田秀司，内平直志，松本一教，本位田真一

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

時制論理ベースの形式的仕様記述言語ＰＴＳ

Temporal Logic based Formal Specification Language

川田秀司　内平直志　松本一教　本位田真一
Hideji Kawata　Naoshi Uchihira　Kazunori Matsumoto　Shinichi Honiden

（株）東芝　システム・ソフトウェア技術研究所
Systems & Software Engineering Lab. TOSHIBA Corp.

あらまし　本論文では，並行プロセスの同期を得るための手法として，形式的仕様記述言語ＰＴＳ（Practical Temporal Specification language)を提案する．これはＰＴＬ(Propositional Temporal Logic)をベースに，各事象を，対象の状態，対象への作用に分け，記述可能にしたものである．また，ＰＴＳの決定手続き，同期手順生成方法，記述例を示す．

１．はじめに

近年，並行プロセスを扱う必要性が増加しているが，このため各々のプロセス間に起こるデッドロックなど同期に関する問題を回避する方法が望まれている．

複数の並行プロセスの同期を得るため，各プロセス間の制約条件をＰＴＬで記述し，タブロー法 [8,7] 等の決定手続きを利用することにより，その制約条件にあった手順を作り出す方法が提案されている[1,2,3,4,5]．しかし，これらの方法で実際の複数プロセス間の制約条件を記述することは難しい．なぜなら，各プロセス間の制約条件は，他の資源の状態に関わることが多い．従ってこれらの状態を記述せず手順を得るためには，あらかじめ対象となるあらゆる資源の変化を明確に把握し，それらを考慮にいれて，制約条件を記述しなければならない．ところが実際にこれらの資源の状態を全て把握することは困難である．

本論文では，上記の問題を解決するために，まず対象（資源など）の状態を記述するのに適した言語ＭＰＴＬ(Many Sorted Propositional Temporal Language)を示す．次に，各プロセス間の制約条件を作用(Action)と，各対象(subject)の状態(Status)とに分けて記述することができるＫＰＴＳ（ Kernel of Practical Temporal Specification language)を提案する．また，このＫＰＴＳをより実践的にしたものとして，ＰＴＳを提案する．これは，ＫＰＴＳを，より現実に即したものにすると同時に，タブローグラフ[6,7]の作成の高速化を可能としたものである．

２．ＰＴＬ

MannaとWolper[1,2]は，通常の命題論理に，時間の概念を導入した Popositional Temporal Logic （ＰＴＬ）を用いて，並列プログラムの同期部を自動生成する研究を行った．ＰＴＬは，通常の命題論理にの記号に加えて，時制オペレータ□，◇，○，Ｕ，を付け加えたものである．

Ｐ，Ｑを論理式とするとき，各時制オペレータは次のような意味を持つ．

□Ｐ　：常にＰが成立．
◇Ｐ　：いつかＰが成立．
○Ｐ　：次の時点で，Ｐが成立．
ＰＵＱ：Ｑが成立するまでは，Ｐが成立．

３．ＭＰＴＬ

状態を表現するのに適した時制論理，ＭＰＴＬを示す．

ＭＰＴＬは，論理式と各基本論理式に対する可能域(Domain)から構成される．

［３．１］ＳＹＮＴＡＸ

[1] ある加算無限集合の元を叙述と呼ぶ．

[2] ｓを任意の叙述とするとき，ｓに対し可能域（Domain of ｓ）と呼ばれる有限集合がただ一つ定まる．

[3] ｓを任意の叙述，Ｄｓをｓの可能域とするとき，ｓ（ｔ）（ｔ∈Ｄｓ）を，基本論理式と呼ぶ．

ＭＰＴＬ論理式は，基本論理式からＰＴＬと同様，帰納的に以下のように構成される．

(A) 基本論理式は，論理式である．

(B) Ａ，Ｂが論理式のとき，

¬Ａ，Ａ∨Ｂ，Ａ∧Ｂ，Ａ⇒Ｂ，Ａ⇔Ｂ，

○Ａ，□Ａ，◇Ａ，ＡＵＢ

は論理式である．

［３．２］ＳＥＭＡＮＴＩＣＳ

[1] 基本論理式ｓ（ｔ）は，各時間において，真（Ｔ）または偽（Ｆ）の値をとる．

[2] 基本論理式ｓ（ｔ）が真となるｔ（∈Ｄｓ）は，各時間において必ず存在し，かつ唯一である．

[3] ¬，∧，∨，⇒，⇔，○，□，◇，Ｕにおける真偽値の決定に関しては，ＰＴＬと同様とする．

また，S⊂DsのときS(S)≡$\bigvee_{t\in S}$s(t)，

と定義する．

これらの定義より明らかに，

$\neg s(t) = \bigvee_{q\in Ds-\{t\}} s(q)$

$\neg s(Q)$　(Q=Ds−{t})

ここで，s(t)=Tの時，「sの値はtである」と呼ぶ．

このMPTL表記法で書かれた任意の論理式は等価なPTL論理式に変換できる．

例）MPTL表記

Domain of h={a, b, c}，

Domain of k={s, t}，

○(h(b)⇒k(t))

をPTLに変換すると，

□((haVhbVhc)∧¬(ha∧hb)

∧¬(hb∧hc)∧¬(hc∧ha)

∧(ksVkt)∧¬(ks∧kt))

∧○(hb⇒kt)

となる．

4．KPTS

複数のプロセス間の制約条件を，いくつかの対象の状態と状態を変化させる作用から構成されるものと考えた時，次の様な性質が考えられる．

・各対象の状態は，作用を受けない限り変化しない．

ここで状態とは，現在の状態と「これ以後どの様な状態をとることができるか」といった，以後の制約を含めたものであると考える．

この性質を，MPTLを使って，次のようにモデル化する．

A) 叙述は対象，叙述の値は状態を表すものとする．

B) 作用は叙述の値を変化させるものとする．

C) 任意の叙述pに対する作用を明確に表現するため，act(p)と表記される特殊な叙述を導入する．

D) 基本論理式act(p)(t)が真であるとは，叙述pの値をtにする作用がその時間に起きたこととする．

E) act(p)は特殊な叙述であり，その値を変化させる作用は存在しない．従って，act(act(p))の様な叙述は，存在しない．

F) act(p)(＊)が真であるとは，pの値を変える作用が存在しない（その時点で，pの値を変える全ての作用が偽）こととする．

G) 叙述pの値がx(∈Dp)とすると，次の時点でact(p)(＊)が真ならば，その時点もpの値は，xでなければならない．

ここで，

act(p)(S)=$\bigvee_{t\in S}$act(p)(t)

(S⊂Domain of act(p))

また，

Domain of act(p)={s|s∈Dp or s=＊}

(Dp；Domain of p)

具体的にいえば，KPTSは，actという特別の意味を持った叙述を導入し，MPTLの各論理式に現れる全ての基本論理式p(t)に対して，下記の二つの関係式を暗黙的に付加した論理である．<u>逆に，KPTSで書かれた論理式に現れる全ての基本論理式に対してそれぞれ上記の関係式を作り，もとの論理式との論理積を取ることにより，MPTLに変換することができる．</u>

□(p(t)∧○act(p)(＊)⇒○p(t))

□(act(p)(t)⇒p(t))．

このことにより，実際の事象の制約を，作用(Action)と各対象の状態(Status)とに分けて記述することが可能となる．

5．PTS

次に，上記のKPTSをより実践的にした，PTSについて述べる．

PTSは，KPTSと次のような違いがある．

1. methodという叙述を導入する．
2. METHOD(m, P, ∧$_i$a$_i$(t$_i$))は宣言で，

「mは，methodの可能域(Domain)の元であり，

□(○method(m)⇒P)

かつ，

□(method(m)

⇒∧$_i$act(a$_i$)(t$_i$)∧Q)

である．」

という意味を持つ（Pは論理式），ただし，Jを使用される全ての叙述の集合とし，S=J−$\bigvee_i$(a$_i$)としたとき，

Q=$\bigwedge_{b\in S}$act(b)(＊)

とする．

3. 作用は，全てmethodによって記述されなければならない．従って，PTSにおいては，act…は，記述するものには，見えなくなる．
4. 記述量を減らすため，各叙述の値に対し変数を使用を許す．変数は，自動的に可能域の値に置換される．（例1参照）
5. 各叙述に対し，n次の配列（有限）が使用でき，また，変数も使用できる（但し，各叙述の可能域は，全て一致しなければならない）．
（例2参照）
6. haltを次のように定める．

halt∈Domain of method．

method(halt)
⇒$\bigwedge_{q \in Q}$act(q)(＊)∧○method(halt)
（Q－{使用される全ての叙述の集合}）
haltは，終了条件の記述に使用する．

7. methodの可能域(Domain)は，
{m_1, m_2, m_3, ‥‥, m_n, halt}，
m_i：METHOD により宣言されたもの．

(例1)
Domain of p = {a,b}, Domain of q = {a,b}
の時，p(A)⇒○q(A)は，
(p(a)⇒○q(a))
∧(p(b)⇒○q(b))
を意味する．

(例2)
次の論理式は，真である．
p[1,1](t)∧p[1,2](s)
∧p[2,1](s)∧p[2,2](t)
∧(p[1,A](s)⇒p[2,A](t))

PTSで書かれた論理式は，KPTSの論理式に変換できる．

6. PTSからの手順の自動生成

PTSは，PTLに変換可能であり，PTLの決定手続きであるタブロー法[6,7]により，手順を生成することができる[1,2,3,4,5]．しかしその変換の過程で論理式の量は数倍に膨れ上がり，タブロー法の計算量は膨大になる．そこで，PTSの論理構造を反映したPTS版タブロー法を新たに開発した．このPTS版タブロー法は，[1]のPTLに対するタブロー法をベースに，次のメカニズムを付加したものである．

(1) methodの値ごとにモデルグラフの辺を作成する．このため，各ノードの分解は，それぞれのmethodの値ごとに行えばよく，また，全体としてノードの数が少なくなるため，分解速度は速くなる．

(2) 叙述pは，act(p)で操作されない限り，前の値を保つ．その対象に対し，act(p)以外の制約が追加された場合は，その状態との積集合をとる．

前の状態:$p(S_1)$ ┐
新しい制約:$p(S_2)$ ┘ ⊢ $p(S_1 \wedge S_2)$

$S_1 \wedge S_2 = \phi$のとき，$p(S_1 \wedge S_2) = F$と判定する．この方法により，OR分解によるノード数の極度の増加を防止することができ，速度は速くなる．

(3) Eventuality のチェックは，グラフを強連結成分に分割しておこなう．これにより，チェックの速度は速くなる．

以上の結果として，生成されるモデルグラフの各辺には，methodの値がただ一つ対応し，モデルグラフ上の遷移により生成できるmethodの値の系列は，PTSの仕様を満す手順となっている．

7. PTSを用いた例題

a，b，c，dの，四つのブロックが2つずつ積まれている．クレーンにはm，nの2つがあり各々独立に動く．

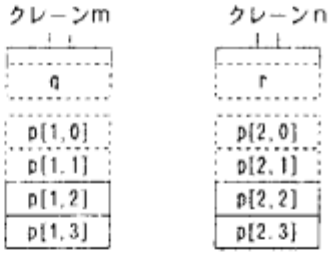
各クレーンの動きを各々3つに分け，

クレーンm
m1：左側のブロックを持ち上げる．
m2：持ち上げたブロックを左側に下ろす．
m3：持ち上げたブロックを右側に下ろす．

クレーンn
n1：右側のブロックを持ち上げる．
n2：持ち上げたブロックを左側に下ろす．
n3：持ち上げたブロックを右側に下ろす．

とする．

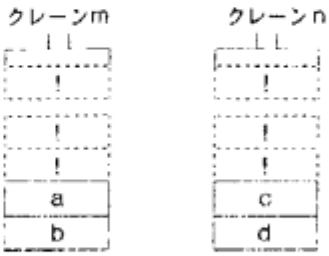
今ブロックの初期状態と終了状態が与えられたとき，これらと状態に対する制約，及びクレーンの動き（作用）をPTSで記述することにより，初期状態から終了状態に移行させる手続きを6.で提案した方法を使って求める．



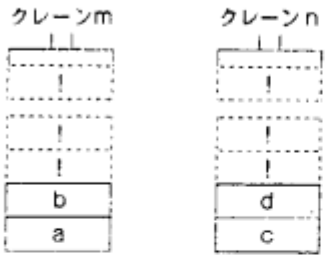
p[i,j] (i∈{1,2},j∈{0,1,2,3}) は，各位置のブロックを表すものとし，Domain of p[i,j]={a,b,c,d,!}（！は，その位置にブロックがないことを示す）とする．またq，rをそれぞれ持ち上げられているブロックの存在を表すものとし，

domain of q = domain of r = {a,b,c,d,!}

また，初期状態を，



とし，終了状態を，



とする．

このとき，ＰＴＳ論理式は次のようになる．

（ｍｅｔｈｏｄ宣言）

METHOD(m1,p[1,A](B)∧¬p[1,A](!)∧q(!)
,q(B)∧p[1,A](!))
METHOD(m2,q(A)∧¬q(!)∧p[1,B](!)
,q(!)∧p[1,B](A))
METHOD(m3,q(A)∧¬q(!)∧p[2,B](!)
,q(!)∧p[2,B](A))
METHOD(n1,p[2,A](B)∧¬p[2,A](!)∧r(!)
,r(B)∧p[2,A](!))
METHOD(n2,r(A)∧¬r(!)∧p[1,B](!)
,r(!)∧p[1,B](A))
METHOD(n3,r(A)∧¬r(!)∧p[2,B](!)
,r(!)∧p[2,B](A))

（終了条件）

∧□(∧p[1,2](b)∧p[1,3](a)
∧p[2,2](d)∧p[2,3](c))
⇒ ○ method(halt))

（ブロック状態の制約）

∧¬(¬p[1,0](!)∧p[1,1](!))
∧¬(¬p[1,1](!)∧p[1,2](!))
∧¬(¬p[1,2](!)∧p[1,3](!))
∧¬(¬p[2,0](!)∧p[2,1](!))
∧¬(¬p[2,1](!)∧p[2,2](!))
∧¬(¬p[2,2](!)∧p[2,3](!))

（初期状態）

∧p[1,0](!)∧p[1,1](!)
∧p[1,2](a)∧p[1,3](b)∧q(!)
∧p[2,0](!)∧p[2,1](!)
∧p[2,2](c)∧p[2,3](d)∧r(!)

（終了状態）

∧◇(∧p[1,2](b)∧p[1,3](a)
∧p[2,2](d)∧p[2,3](c))

このＰＴＳによる記述は，従来のＰＴＬによる記述の約１／５であり，手順作成速度もＰＴＳ版タブロー法により，大幅に向上した．

8．終わりに

今回，並行プロセスの同期の仕様を記述するための，ＰＴＳについて述べた．

このＰＴＳの導入により，記述し易さを高めることができ，また記述力が同等であるＰＴＬの決定手続きに対し，処理能力（処理時間）を向上させることができた．

今後さらに，

1) ＰＴＳ（ＫＰＴＳ）の記述力（クラス）を上げる．
2) 決定手続きの高速化．

をはかる予定である．

謝辞

本研究の一部は，第５世代コンピュータプロジェクトの一環として行われたものであり，研究の機会を与えて下さった新世代コンピュータ技術開発機構の方々感謝する．また日頃ご指導いただいている，（株）東芝 システム・ソフトウェア技術研究所，中村英夫主任研究員に深謝する．

参考文献

[1] Manna,Z. and Walper,P. "Synthesis of Communicating Processes from Temporal Logic Specifications", Report STAN-CS-81-872, Stanford University Computer Science Dept.,September 1981.

[2] Wolper,P."Synthesis of Communicating Processes from Temporal Logic Specifications",STAN-CS-82-925,Stanford University ,1982.

[3] 内平，本位田：時制命題論理を用いた部品からの並列プログラム合成，日本ソウトウェア科学会第３回大会論文集，pp 145-148,1986.

[4] Stuart,C. "AN IMPLEMENTATION OF A MULTI-AGENT PLAN SYNCHRONIZER", IJCAI-85,pp 1031-1033,1985.

[5] Clarke,E.M and Emerson,E.A. "Design and synthesis of synchronization skeletons using branching time temporal logic" , Logics of programs (Proceedings 1981), Lecture Note in Computer Science 131,Springer-Verlag,pp 52-72, 1982.

[6] Plaisted,D.A. "A Decision Procedure for Combinations of Propositional Temporal Logic and Other Specialized Theories",Journal of Automated Reasoning 2,pp 171-190,1986.

[7] Smullyan,R.M. "First-Order Logic" Springer-Verlag,Heidelberg,1968