発表番号：1　　2007 年 01 月 29 日

時間付き通信プロセスモデルにおける時間動作の抽象化

桑原 寛明　結縁 祥治　阿草 清滋
名古屋大学大学院情報科学研究科

概 要

本稿では，時間拡張した π 計算のプロセスを時間に関して抽象化する手法を提案する．提案手法は，時間動作も考慮した双模倣関係に基づく展開定理によるプロセス式の変形と，時間経過動作の抽象化からなる．時間動作を抽象化したプロセスにおける双模倣関係と，入出力動作とタイムアウトのみに着目した双模倣関係が等価であることを示す．時間動作の抽象化により，入出力動作のみに着目してプロセスの振舞いを解析することができる．

1　はじめに

実時間システムのソフトウェアは，時間制約を持つ複数のコンポーネントを組み合わせ，各コンポーネントが状態を持ちながら並行に動作するプログラムとして構成される．その上で，システム全体として時間制約を守ることが要求される．それぞれのコンポーネントの動作には時間制約が存在し，さらに並行に動作する他のコンポーネントの影響を受けるため，このような実時間並行プログラムの振舞い解析は逐次プログラムに比べて難しい．

筆者らは，実時間システムの振舞い解析を目的として，時間に関する動作も含めてシステムの振舞いを詳細に記述するために π 計算の時間拡張を提案した [2, 3]．提案した体系に基づいて等価関係と擬順序関係を定義し，合同的性質を持つ十分条件を示した．時間待ちやタイムアウトを直接的に記述可能であり代数的合同性が示されているため，時間制約が存在する振舞いの性質を解析する基礎技法とできる．

我々が提案した体系では時間経過を細かく追跡することが可能なため，経過時間の長さや動作のタイミングが重要な意味を持つ性質の解析に適している．しかし，経過時間の長さに依存しない性質などに対しては適しているとはいえない．時間をモデル化して状態を細かく分類するため，状態爆発の問題が深刻になるからである．示したい性質を保存したまま時間に関して抽象化を行い，状態数を減らす手法が必要である．

π 計算 [5, 4, 7] は高い表現能力を持つ並行システムの抽象計算モデルであり，代数的性質に基づく様々は技法が適用できる．プロセスの双模倣性の判定 [6] やモデル検査 [1, 8] について多くの研究がなされており，Mobility Workbench[11] といったプロセスのシミュレートや等価性判定を行うツールも作成されている．時間に関する動作を抽象化することによりこれら既存の手法や実装が適用可能になり有用である．

本稿では時間拡張した π 計算のプロセスを時間に関して抽象化する手法を提案する．提案手法は，時間付き双模倣関係によるプロセス式の変形と，時間経過動作の抽象化からなる．プロセス式の変形を容易にするために時間拡張した π 計算における展開定理を示す．時間経過動作を抽象化することで入力や出力動作に関する振舞いのみが解析できる．

2　時間付き π 計算

2.1　構文と動作意味

文献 [3] に基づき時間付き π 計算の構文と動作意味を示す．π 計算に自然数によって添字付けされたプレフィックス t を導入する．m 単位時間の長さの時間待ちを $t[\underline{m}]$ と記述し，時間経過動作と呼ぶ．ここで t は以下に示す名前には含まれない特別なシンボルとする．

$\mathcal{N}ame$ を名前の集合，$\mathcal{I}$ を自然数を表す名前の集合，$\mathbb{N}$ を自然数の集合とし，$\mathcal{N} = \mathcal{N}ame - \mathcal{I}$ とする．ここで，$m \in \mathcal{I}$ に対し $i_m \in \mathbb{N}$ が存在して m は $\underline{i_m}$ と書ける．$\mathcal{I}$ 上の演算 $+, \dot{-}, <, \leq$ を定義する．

定義 1 $i_m, i_n \in \mathbb{N}$，$\underline{i_m}, \underline{i_n} \in \mathcal{I}$ に対し

$$\underline{i_m} + \underline{i_n} \stackrel{def}{=} \underline{i_m + i_n}$$
$$\underline{i_m} \dot{-} \underline{i_n} \stackrel{def}{=} \begin{cases} \underline{i_m - i_n} & \text{if } i_m \geq i_n \\ \underline{0} & \text{otherwise} \end{cases}$$
$$\underline{i_m} < \underline{i_n} \stackrel{def}{=} i_m < i_n$$
$$\underline{i_m} \leq \underline{i_n} \stackrel{def}{=} i_m \leq i_n \qquad \square$$

時間付き π 計算のプロセス全体の集合を $\mathcal{P}$ と書く．Act を動作 $x(y), \overline{x}\langle z\rangle, \overline{x}(z), \tau$ 全体の集合とする．$\overline{x}(z)$ は $(\boldsymbol{\nu} z)\,\overline{x}\langle z\rangle$ の略記であり，新しい名前 z を x を通して出力する動作を表す．以下では，$n, x, y, z \in \mathcal{N}ame$, $d \in \mathcal{I}$，$\alpha, \beta \in Act$ とする．

定義 2 時間付き π 計算のプロセス式 P は以下の構文によって定義される．

$$\begin{array}{lcl} \pi & ::= & x(y)@d \mid \overline{x}\langle z\rangle@d \mid \tau@d \mid t[n] \mid [x=y]\pi \\ P & ::= & M \mid P \mid P \mid \boldsymbol{\nu} x\, P \mid !\, P \\ M & ::= & \mathbf{0} \mid \pi.P \mid M + M \end{array} \qquad \square$$

定義 3 プロセス $x(y)@d.P$ における名前 y, d, $\overline{x}\langle z\rangle@d.P$，$\tau@d.P$ における名前 d，$\boldsymbol{\nu} x\, P$ における名前 x のスコープは P に制限される．この時，名前 y, d, x は束縛されるという．束縛されない名前を自由であるという．プロセス P に含まれる自由な名前の集合を $\mathsf{fn}(P)$，束縛される名前の集合を $\mathsf{bn}(P)$ と書く．$\mathsf{n}(\alpha)$ を動作 α に出現する名前の集合とする． $\square$

$x(y)@d.P$ や $\overline{x}\langle z\rangle@d.P$，$\tau@d.P$ において $d \notin \mathsf{fn}(P)$ の場合，それぞれ $x(y).P$，$\overline{x}\langle z\rangle.P$，$\tau.P$ とも書く．時間経過動作 $t[n]$ の n も入力プレフィックスや制限演算子によって束縛される．次に名前に対する代入を定義する．

定義 4 代入は $\mathcal{N}ame$ から $\mathcal{N}ame$ への関数である．プロセス P に代入 σ を適用して得られるプロセス $P\sigma$ を以下のように定義する．ここで，$x\sigma$ は名前 x に σ を適用して得られる名前を表し，σ_{-N} は集合 N に含まれる名前については置換を行わず，その他の名前については σ と同じ置換を行う代入を表す．

$$\begin{array}{ll} \mathbf{0}\sigma & \stackrel{def}{=} \mathbf{0} \\ (x(y)@d.P)\sigma & \stackrel{def}{=} x\sigma(y)@d.P\sigma_{-\{y,d\}} \\ (\overline{x}\langle z\rangle@d.P)\sigma & \stackrel{def}{=} \overline{x\sigma}\langle z\sigma\rangle@d.P\sigma_{-\{d\}} \\ (\tau@d.P)\sigma & \stackrel{def}{=} \tau.P\sigma_{-\{d\}} \\ (t[n].P)\sigma & \stackrel{def}{=} t[n\sigma].P\sigma \\ ([x=y]\pi.P)\sigma & \stackrel{def}{=} [x\sigma = y\sigma]\pi\sigma.P\sigma \\ (P \mid Q)\sigma & \stackrel{def}{=} P\sigma \mid Q\sigma \\ (P+Q)\sigma & \stackrel{def}{=} P\sigma + Q\sigma \\ (\boldsymbol{\nu} x\, P)\sigma & \stackrel{def}{=} \boldsymbol{\nu} x\, P\sigma_{-\{x\}} \\ (!\, P)\sigma & \stackrel{def}{=} !\, P\sigma \end{array} \qquad \square$$

代入 σ は $\{y_1, \ldots, y_n/x_1, \ldots, x_n\}$ とも書く．これは x_i に対する y_i の代入を表す．

$\mathcal{P}$ 上の遷移関係 $\{\stackrel{\alpha}{\rightarrow} \mid \alpha \in Act \cup \{\bullet\}\} \cup \{\rightarrow\}$ を図 1 の遷移規則及び図 2 の時間経過規則によって定義する．ここで $\bullet$ は Act に含まれない特別なシンボルとする．図 1 では T-SUM-L 規則，SUM-L 規則，T-PAR-L 規則，PAR-L 規則，COMM-L 規則，CLOSE-L 規則においてそれぞれ P と Q を入れ替えた T-SUM-R 規則，SUM-R 規則，T-PAR-R 規則，PAR-R 規則，COMM-R 規則，CLOSE-R 規則が省略されている．RES 規則，REP-ACT 規則においては $\alpha = \bullet$ の場合も含む．

時間経過動作 $t[\underline{k}]$ は k 単位時間待機することを表す．例えば，プロセス $t[\underline{10}].P$ は 10 単位時間待機した後にプロセス P として振舞う．タイムアウトは $a.P + t[\underline{5}].Q$ のように時間経過動作と選択演算子を用いて記述できる．このプロセスは a 動作の発生を最大で 4 単位時間待機する．4 単位時間以内に a を実行すれば P へ遷移する．5 単位時間経過した時にタイムアウトし Q へ遷移する．

$P \stackrel{\alpha}{\rightarrow} P'$(ただし$\alpha \neq \bullet$) は入力，出力，内部動作のいずれかの動作 α により P から P' に遷移することを表し，$P \rightarrow P'$ は P が 1 単位時間の経過により P' に遷移することを表す．$\mathrm{PAR_T}$ 規則と $\mathrm{REP_T}$ 規則は τ 動作による遷移が時間経過による遷移に優先して発生することを表す．

$P \stackrel{\bullet}{\rightarrow} P'$ はタイムアウトによって P から P' に遷移することを表し，入出力動作や時間経過に優先して実行される．$+$ はタイムアウトによって選択が行われる．例えば，$a.P + t[\underline{0}].Q \stackrel{\bullet}{\rightarrow} Q$ である．$t[\underline{3}].P + t[\underline{5}].Q$ では $t[\underline{3}].P$ が先にタイムアウトするため必ず $t[\underline{3}].P$ が選

$$\text{OUT}: \frac{}{\overline{x}\langle z\rangle@d.P \xrightarrow{\overline{x}\langle z\rangle} P\{\underline{0}/d\}} \qquad \text{IN}: \frac{}{x(y)@d.P \xrightarrow{x(z)} P\{z/y\}\{\underline{0}/d\}}$$

$$\text{TAU}: \frac{}{\tau@d.P \xrightarrow{\tau} P\{\underline{0}/d\}} \qquad \text{TIMEOUT}: \frac{}{t[\underline{0}].P \xrightarrow{\bullet} P}$$

$$\text{MATCH}: \frac{\pi.P \xrightarrow{\alpha} P'}{[x=x]\pi.P \xrightarrow{\alpha} P'}$$

$$\text{T-SUM-L}: \frac{P \xrightarrow{\bullet} P'}{P+Q \xrightarrow{\bullet} P'} \qquad \text{SUM-L}: \frac{P \xrightarrow{\alpha} P' \quad Q \overset{\bullet}{\not\rightarrow}}{P+Q \xrightarrow{\alpha} P'}\ \alpha \neq \bullet$$

$$\text{T-PAR-L}: \frac{P \overset{\bullet}{\rightarrow} P'}{P \mid Q \overset{\bullet}{\rightarrow} P' \mid Q}$$

$$\text{PAR-L}: \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q}\ \alpha \neq \bullet \wedge ((z \in \mathcal{I} \vee z \notin \mathsf{fn}(Q)) \text{ if } \alpha = \overline{x}(z))$$

$$\text{COMM-L}: \frac{P \xrightarrow{\overline{x}\langle z\rangle} P' \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \text{CLOSE-L}: \frac{P \xrightarrow{\overline{x}(z)} P' \quad Q \xrightarrow{x(z)} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu} z\,(P' \mid Q')}\ z \notin \mathsf{fn}(Q)$$

$$\text{RES}: \frac{P \xrightarrow{\alpha} P'}{\boldsymbol{\nu} x\,P \xrightarrow{\alpha} \boldsymbol{\nu} x\,P'}\ x \notin \mathsf{n}(\alpha) \qquad \text{OPEN}: \frac{P \xrightarrow{\overline{x}\langle z\rangle} P'}{\boldsymbol{\nu} z\,P \xrightarrow{\overline{x}(z)} P'}\ z \neq x$$

$$\text{REP-ACT}: \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P} \qquad \text{REP-COMM}: \frac{P \xrightarrow{\overline{x}\langle z\rangle} P' \quad P \xrightarrow{x(z)} P''}{!P \xrightarrow{\tau} (P' \mid P'') \mid !P}$$

$$\text{REP-CLOSE}: \frac{P \xrightarrow{\overline{x}(z)} P' \quad P \xrightarrow{x(z)} P''}{!P \xrightarrow{\tau} (\boldsymbol{\nu} z\,(P' \mid P'')) \mid !P}\ z \notin \mathsf{fn}(P)$$

図 1: 遷移規則

択され，3 単位時間後に P に遷移する．

他のプレフィックスと演算子の直観的な意味を以下に示す．

- （入力）$x(y)$：x を通して y に受信
- （出力）$\overline{x}\langle z\rangle$：$x$ を通して z を送信
- （τ 動作）τ：外部からは不可視な内部アクション
- （並行）$P \mid Q$：プロセス P, Q が並行に動作する
- （制限）$\boldsymbol{\nu} x\,P$：プロセス P 中の自由な変数 x を束縛する
- （複製）$!P$：無限個のプロセス P が $\mid$ 演算子によって結合しているとみなす

2.2 時間的性質

時間付き π 計算のプロセスは可能動作不変性 (Constancy of Offers)[10] を満たす．

定理 1 (**可能動作不変性**) 任意のプロセス P に対し，$P \rightharpoonup P'$ かつ $P' \overset{\bullet}{\not\rightarrow}$ ならば $P \overset{\alpha}{\rightarrow} \iff P' \overset{\alpha}{\rightarrow}$ である．

証明： $P \rightharpoonup P'$ の導出木の高さに関する帰納法による．導出の最後に適用された規則によって場合分けする．

- PASS_T 規則：$P = t[\underline{k}].P_0$ (ただし $k > 0$) であり，$P' = t[\underline{k-1}].P_0 \overset{\bullet}{\not\rightarrow}$ であることから $k - 1 > 0$，すなわち $k > 1$ である．この時，遷移規則の定義より明らかに $P \overset{\alpha}{\not\rightarrow}$ かつ $P' \overset{\alpha}{\not\rightarrow}$ である．
- INACT_T 規則：$P = P' = \mathbf{0}$ ゆえ明らかに $P \overset{\alpha}{\not\rightarrow}$ かつ $P' \overset{\alpha}{\not\rightarrow}$ である．
- OUT_T 規則，IN_T 規則：$P = P'$ ゆえ明らかに $P \overset{\alpha}{\rightarrow}$ ならば $P' \overset{\alpha}{\rightarrow}$ かつ $P' \overset{\alpha}{\rightarrow}$ ならば $P \overset{\alpha}{\rightarrow}$ である．
- SUM_T 規則：$P = P_1 + P_2, P_1 \rightharpoonup P_1', P_2 \rightharpoonup P_2'$, $P' = P_1' + P_2'$ とする．$P \overset{\bullet}{\not\rightarrow}$ ゆえ SUM-L, SUM-R 規則から $P_1' \overset{\bullet}{\not\rightarrow}$ かつ $P_2' \overset{\bullet}{\not\rightarrow}$ である．ここで，帰納法

$$\text{PASS}_{\text{T}}:\ \frac{}{t[\underline{k}].P \rightharpoonup t[\underline{k-1}].P}\ \text{if } k>0 \qquad \text{INACT}_{\text{T}}:\ \frac{}{\mathbf{0} \rightharpoonup \mathbf{0}}$$

$$\text{OUT}_{\text{T}}:\ \frac{}{\overline{x}\langle z\rangle@d.P \rightharpoonup \overline{x}\langle z\rangle@d.P\{d+\underline{1}/d\}}$$

$$\text{IN}_{\text{T}}:\ \frac{}{x(y)@d.P \rightharpoonup x(y)@d.P\{d+\underline{1}/d\}}$$

$$\text{N-MAT}_{\text{T}}:\ \frac{}{[x=y]\pi.P \rightharpoonup [x=y]\pi.P}\ x\neq y \qquad \text{P-MAT}_{\text{T}}:\ \frac{\pi.P \rightharpoonup P'}{[x=x]\pi.P \rightharpoonup P'}$$

$$\text{SUM}_{\text{T}}:\ \frac{P \rightharpoonup P' \quad Q \rightharpoonup Q'}{P+Q \rightharpoonup P'+Q'} \qquad \text{PAR}_{\text{T}}:\ \frac{P \rightharpoonup P' \quad Q \rightharpoonup Q'}{P \mid Q \rightharpoonup P' \mid Q'}\ \text{if } P \mid Q \overset{\tau}{\not\rightharpoonup}$$

$$\text{RES}_{\text{T}}:\ \frac{P \rightharpoonup P'}{\boldsymbol{\nu} x\, P \rightharpoonup \boldsymbol{\nu} x\, P'} \qquad \text{REP}_{\text{T}}:\ \frac{P \rightharpoonup P'}{!P \rightharpoonup !P'}\ \text{if } P \mid P \overset{\tau}{\not\rightharpoonup}$$

図 2: 時間経過規則

の仮定から $P_1 \overset{\alpha}{\rightharpoonup} \iff P_1' \overset{\alpha}{\rightharpoonup}$ および $P_2 \overset{\alpha}{\rightharpoonup} \iff P_2' \overset{\alpha}{\rightharpoonup}$ であるため，$P \overset{\alpha}{\rightharpoonup} \iff P' \overset{\alpha}{\rightharpoonup}$ である．

- 他の場合も同様. □

可能動作不変性は，時間経過後にタイムアウトしない場合，時間経過前後で実行できる入出力動作が変化しないことを表す.

2.3 双模倣関係

文献 [3] における時間付き π 計算の双模倣関係はタイムアウトを抽象した遷移関係に基づいている．そのため，タイムアウトに着目してプロセスの振舞いを解析することはできない．時間動作の抽象化をタイムアウトに着目して行うため，タイムアウトを抽象しない双模倣関係を定義する.

定義 5 以下を満たす対称な関係 $\mathcal{R}$ を時間付き詳細双模倣関係と呼ぶ.

$(P,Q) \in \mathcal{R}$ の時,

- $P \overset{\alpha}{\rightharpoonup} P' \ \Rightarrow\ \exists Q'.\ Q \overset{\alpha}{\rightharpoonup} Q' \wedge (P',Q') \in \mathcal{R}$
- $P \rightharpoonup P'' \ \Rightarrow\ \exists Q''.\ Q \rightharpoonup Q'' \wedge (P'',Q'') \in \mathcal{R}$ □

$(P,Q) \in \mathcal{R}$ なる時間付き詳細双模倣関係 $\mathcal{R}$ が存在する時 $P \sim_T^f Q$ と書く.

定義 6 $\sim_T^f\ =\ \bigcup\{\mathcal{R} \mid \mathcal{R}$ は時間付き詳細双模倣関係 $\}$ □

命題 1 $\sim_T^f$ は最大の時間付き詳細双模倣関係である. □

命題 2 $\sim_T^f$ は等価関係である. □

時間付き詳細双模倣関係は入力プレフィックスに対して保存されない．入力プレフィックスを含むすべての演算子に対して保存されるような時間付き詳細双模倣関係の部分集合を求める.

定義 7 任意の代入 σ に対して $P\sigma \sim_T^f Q\sigma$ であるような P と Q の関係を時間付き完全詳細双模倣関係と呼び，$P \sim_T^{fc} Q$ と書く. □

定義より明らかに $\sim_T^{fc} \subseteq \sim_T^f$ である.

定義 8 コンテキスト $C[\cdot]$ を以下のように定義する.

$$\begin{aligned} C[\cdot] \ ::=\ & [\cdot] \mid \pi.C[\cdot] \mid \pi.C[\cdot]+R \mid C[\cdot] \mid S \\ & \mid \boldsymbol{\nu} x\, C[\cdot] \mid\ !C[\cdot] \end{aligned}$$

ここで R はガード付きプロセス，S は任意のプロセスとする. □

定理 2 $P \sim_T^{fc} Q$ の時，任意のコンテキスト $C[\cdot]$ に対し $C[P] \sim_T^{fc} C[Q]$ である. □

3 展開定理

時間付き詳細双模倣関係のもとで成り立つ展開定理は，複数のサブプロセスの並行合成によって構成されるプロセスに対して，振舞い等価な単一のプロセスが存在することを表す.

定義 2 の構文の定義と並行合成演算子に関する構造合同関係から，並行合成されるプロセスは一般的に

$\sum_{i\in I} M_i\alpha_i@d_i.P_i + \sum_{j\in J} N_j t[n_j].Q_j$ と表すことができる．ここで，$I = \{1,\ldots,n\}$ とする時 $\sum_{i\in I} P_i$ は $P_1+\cdots+P_n$ の略記である．M_i, M'_i, N_j, N'_j は比較演算 $[x_1 = y_1][x_2 = y_2]\cdots[x_n = y_n]$ を表す．すべての $i \in \{1\ldots n\}$ に対して $x_i = y_i$ の時 true，true でなければ false と書く．

定理 3 $P = \sum_i M_i\alpha_i@d_i.P_i + \sum_i M'_i t[n_i].P'_i$，$Q = \sum_j N_j\beta_j@e_j.Q_j + \sum_j N'_j t[m_j].Q'_j$ とする．この時，

$$
\begin{aligned}
&P \,|\, Q \\
&\sim^{fc}_{\mathcal{T}} \quad \textstyle\sum_i M_i\alpha_i@d_i.(P_i \,|\, \sum_j N_j\beta_j@e_j.Q_j\{e_j{+}d_i/e_j\} \\
&\qquad\qquad + \textstyle\sum_j N'_j t[m_j \dot{-} d_i].Q'_j) \\
&\quad + \textstyle\sum_j N_j\beta_j@e_j.(\sum_i M_i\alpha_i@d_i.P_i\{d_i{+}e_j/d_i\} \\
&\qquad\qquad + \textstyle\sum_i M'_i t[n_i \dot{-} e_j].P'_i \,|\, Q_j) \\
&\quad + \textstyle\sum_i \sum_j M_i N_j[x = y]\tau.R_{ij} \\
&\quad + \textstyle\sum_i M'_i t[n_i].(P'_i \,|\, \sum_j N_j\beta_j@e_j.Q_j\{e_j{+}n_i/e_j\} \\
&\qquad\qquad + \textstyle\sum_j N'_j t[m_j \dot{-} n_i].Q'_j) \\
&\quad + \textstyle\sum_j N'_j t[m_j].(\sum_i M_i\alpha_i@d_i.P_i\{d_i{+}m_j/d_i\} \\
&\qquad\qquad + \textstyle\sum_i M'_i t[n_i \dot{-} m_j].P'_i \,|\, Q'_j)
\end{aligned}
$$

ここで R_{ij} は α_i, β_j の組み合わせによる以下のいずれかである．

α_i	β_j	R_{ij}
$\overline{x}\langle v\rangle$	$y(u)$	$P_i\{\underline{0}/d_i\} \,\|\, Q_j\{v/u\}\{\underline{0}/e_j\}$
$\overline{x}(v)$	$y(v)$	$(\boldsymbol{\nu} v)\,(P_i\{\underline{0}/d_i\} \,\|\, Q_j\{\underline{0}/e_j\})$
$x(u)$	$\overline{y}\langle v\rangle$	$P_i\{v/u\}\{\underline{0}/d_i\} \,\|\, Q_j\{\underline{0}/e_j\}$
$x(v)$	$\overline{y}(v)$	$(\boldsymbol{\nu} v)\,(P_i\{\underline{0}/d_i\} \,\|\, Q_j\{\underline{0}/e_j\})$

証明：(スケッチ)

$$
\begin{aligned}
\mathcal{R} \;=\; &\{(P(l) \,|\, Q(l), \\
&\quad S_{P_1}(l) + S_{Q_1}(l) + S_R(l) + S_{P_2}(l) + S_{Q_2}(l)) \\
&\quad \mid \underline{0} \le l \le l_\tau\} \cup Id
\end{aligned}
$$

に対し $\mathcal{R} \subseteq \sim^{fc}_{\mathcal{T}}$ であることを示せばよい．ここで，

$$
\begin{aligned}
P(l) \;&=\; \textstyle\sum_i M_i\alpha_i@d_i.P_i\{d_i + l/d_i\} \\
&\quad + \textstyle\sum_i M'_i t[n_i \dot{-} l].P'_i \\
Q(l) \;&=\; \textstyle\sum_j N_j\beta_j@e_j.Q_j\{e_j + l/e_j\} \\
&\quad + \textstyle\sum_j N'_j t[m_j \dot{-} l].Q'_j \\
S_{P_1}(l) \;&=\; \textstyle\sum_i M_i\alpha_i@d_i.(P_i\{d_i{+}l/d_i\} \\
&\qquad \mid \textstyle\sum_j N_j\beta_j@e_j.Q_j\{e_j{+}d_i{+}l/e_j\} \\
&\qquad + \textstyle\sum_j N'_j t[m_j \dot{-} d_i \dot{-} l].Q'_j) \\
S_{Q_1}(l) \;&=\; \textstyle\sum_j N_j\beta_j@e_j \\
&\qquad .(\sum_i M_i\alpha_i@d_i.P_i\{d_i{+}e_j{+}l/d_i\} \\
&\qquad + \textstyle\sum_i M'_i t[n_i \dot{-} e_j \dot{-} l].P'_i \\
&\qquad \mid Q_j\{e_j{+}l/e_j\}) \\
S_R(l) \;&=\; \textstyle\sum_i \sum_j M_i N_j[x = y]\tau.R_{ij} \\
S_{P_2}(l) \;&=\; \textstyle\sum_i M'_i t[n_i \dot{-} l].(P'_i \\
&\qquad \mid \textstyle\sum_j N_j\beta_j@e_j.Q_j\{e_j{+}n_i/e_j\} \\
&\qquad + \textstyle\sum_j N'_j t[m_j \dot{-} n_i].Q'_j) \\
S_{Q_2}(l) \;&=\; \textstyle\sum_j N'_j t[m_j \dot{-} l] \\
&\qquad .(\sum_i M_i\alpha_i@d_i.P_i\{d_i{+}m_j/d_i\} \\
&\qquad + \textstyle\sum_i M'_i t[n_i \dot{-} m_j].P'_i \,|\, Q'_j)
\end{aligned}
$$

$$
l_\tau \;=\; \begin{cases}
\underline{0} & \text{if } \exists i.(M_i = \mathsf{true} \wedge \alpha_i = \tau) \\
& \quad \vee \exists j.(N_j = \mathsf{true} \wedge \beta_j = \tau) \\
& \quad \vee \exists i,j.(M_i = \mathsf{true} \wedge N_j = \mathsf{true} \\
& \qquad \wedge\, x = y) \\
\min(\{n_i\} \cup \{m_j\}) & \text{otherwise}
\end{cases}
$$

である．ただし $\min(\emptyset) = \infty$ とする． □

複製演算子に対する展開定理を以下に示す．

定理 4 $P = \sum_i M_i\alpha_i@d_i.P_i + \sum_i M'_i t[n_i].P'_i$ とする．この時，

$$
\begin{aligned}
!P \quad &\sim^{fc}_{\mathcal{T}} \quad \textstyle\sum_i M_i\alpha_i@d_i.(P_i \\
&\qquad\qquad \mid !\,(\textstyle\sum_j M_j\alpha_j@d_j.P_j\{d_j + d_i/d_j\} \\
&\qquad\qquad + \textstyle\sum_j M'_j t[n_j \dot{-} d_i].P'_j)) \\
&\quad + \textstyle\sum_i \sum_j M_i M_j[x = y]\tau.(R_{ij} \,|\, !P) \\
&\quad + \textstyle\sum_i M'_i t[n_i].(P'_i \\
&\qquad\qquad \mid !\,(\textstyle\sum_j M_j\alpha_j@d_j.P_j\{d_j + n_i/d_j\} \\
&\qquad\qquad + \textstyle\sum_j M'_j t[n_j \dot{-} n_i].P'_j))
\end{aligned}
$$

ここで R_{ij} は α_i, α_j の組み合わせによる以下のいずれかである．

α_i	α_j	R_{ij}
$\overline{x}\langle v\rangle$	$y(u)$	$P_i\{\underline{0}/d_i\} \,\|\, P_j\{v/u\}\{\underline{0}/e_j\}$
$\overline{x}(v)$	$y(v)$	$(\boldsymbol{\nu} v)\,(P_i\{\underline{0}/d_i\} \,\|\, P_j\{\underline{0}/e_j\})$
$x(u)$	$\overline{y}\langle v\rangle$	$P_i\{v/u\}\{\underline{0}/d_i\} \,\|\, P_j\{\underline{0}/e_j\}$
$x(v)$	$\overline{y}(v)$	$(\boldsymbol{\nu} v)\,(P_i\{\underline{0}/d_i\} \,\|\, P_j\{\underline{0}/e_j\})$

□

4 時間経過動作の抽象化

4.1 並行合成の展開

時間経過動作の抽象化は，展開定理に基づいて元のプロセス式の並行合成演算子と複製演算子を展開して得られるプロセス式に対して行う．展開定理から展開前後のプロセスは時間付き詳細双模倣である．

定義 9 時間付き π 計算のプロセス式に対し並行合成演算子および複製演算子の展開を行う関数 $[\![\cdot]\!]_e$ を図 *3* のように定義する．ただし，$[\![P_a]\!]_e = \sum_i M_i\alpha_i@d_i.P_i + \sum_i M'_i t[n_i].P'_i$ および $[\![P_b]\!]_e = \sum_j N_j\beta_j@e_j.Q_j + \sum_j N'_j t[m_j].Q'_j$ とする．また R_{ij} は定理 *3* における R_{ij}，R'_{ij} は定理 *4* における R_{ij} である． □

補題 1 任意のプロセス式 $P \in \mathcal{P}$ に対し $P \sim^{fc}_{\mathcal{T}} [\![P]\!]_e$ である． □

4.2 時間経過動作の抽象

タイムアウトが発生してから次のタイムアウトが発生するまでの時間経過を抽象する．例えば，$a.P+t[\underline{5}].Q$ は 4 単位時間以内に動作 a が発生すれば P へ，動作 a が発生せず 5 単位時間経過すればタイムアウト動作を行って Q へ遷移する．この時，実行開始から 4 単位時間経過するまでは動作 a を実行するか時間経過するかのいずれかの可能性しか存在しない．そこで，$a.P + t.Q$ のように時間経過動作によるタイムアウトを通常の動作 t として抽象化することを考える．Q へ遷移するまでに経過する時間の長さはわからなくなるが，動作 a によって P へ遷移するか，あるいは時間経過によって Q へ遷移することは表現されている．

展開定理により プロセス式は一般的に $(\boldsymbol{\nu}\tilde{z})\,(\sum_i M_i\alpha_i@d_i.P_i + \sum_j N_j t[n_j].Q_j)$ のように逐次プロセスの形で表現可能である．そこで，逐次プロセスを対象として時間経過動作の抽象化を定義する．

定義 10 並行プロセスではない単一のプロセスを表すプロセス式 $P \in \mathcal{P}$ に対し時間経過動作の抽象化 $[\![P]\!]_t$ を以下のように定義する．

$$
\begin{aligned}
&[\![\mathbf{0}]\!]_t \stackrel{def}{=} \mathbf{0}\\
&[\![\boldsymbol{\nu}x\,P]\!]_t \stackrel{def}{=} \boldsymbol{\nu}x\,[\![P]\!]_t\\
&[\![\textstyle\sum_{i\in I} M_i\alpha_i@d_i.P_i + \sum_{j\in J} N_j t[n_j].Q_j]\!]_t\\
&\quad\stackrel{def}{=}\begin{cases}
\sum_{j\in J'} N_j t.Q_j \\
\qquad \text{if } \exists j.(N_j = \mathsf{true} \wedge n_j = \underline{0})\\
\sum_{i\in I} M_i\alpha_i.P_i\{\underline{0}/d_i\}\\
\qquad \text{if } \forall j.(N_j = \mathsf{false} \vee \underline{0} < n_j) \wedge\\
\qquad\quad \exists i.(M_i = \mathsf{true} \wedge \alpha_i = \tau)\\
\sum_{i\in I} M_i\alpha_i.P_i\{\underline{0}/d_i\} + \sum_{j\in J''} N_j t.Q_j\\
\qquad \text{otherwise}
\end{cases}
\end{aligned}
$$

ここで $J' = \{j' \,|\, N_{j'} = \mathsf{true} \wedge n_{j'} = \underline{0}\}$，$J'' = \{j' \,|\, n_{j'} = \min(\{n_j \,|\, N_j = \mathsf{true}\})\}$ である．また，t はタイムアウトを表す名前であり，P および $[\![P]\!]_t$ には出現しないとする． □

時間経過動作の抽象化はプロセス式のプレフィックスのみに対して適用され，サブプロセス式に対しては再帰的に適用されない．時間経過動作の添字が入力プレフィックスによって束縛される場合，どのように抽象化するか決定できない可能性がある．例えば，プロセス式 $x(n).(t[n].Q + t[\underline{5}].R)$ において $t[n].Q + t[\underline{5}].R$ も抽象化しようとしても，入力動作 $x(m)$ によって n に代入される名前 m に応じて抽象化は以下の 3 通りの可能性が存在し，一意に決めることができない．

$$
\begin{array}{ll}
t.R & \text{if } m < \underline{5}\\
t.Q + t.R & \text{if } m = \underline{5}\\
t.Q & \text{if } \underline{5} < m
\end{array}
$$

よって，$t[n].Q + t[\underline{5}].R$ を精密に抽象化するためには入力動作が実行されて n に代入される名前が確定するまで待つ必要がある．入出力動作，τ 動作，時間経過動作を抽象化した t 動作が実行されるごとに抽象化を行う．

並行合成の展開と逐次プロセスに対する時間経過動作の抽象化を組み合わせて，任意のプロセスに対する時間経過動作の抽象化を定義する．

定義 11 プロセス式 $P \in \mathcal{P}$ に対し時間経過動作の抽象化 $[\![P]\!]_a$ を

$$[\![P]\!]_a \stackrel{def}{=} [\![[\![P]\!]_e]\!]_t$$

と定義する． □

4.3 性質

時間経過動作を抽象する前後のプロセス間の関係について述べる．初めに時間経過動作を抽象化するプロセスの間の関係を定義する．

$$
\begin{aligned}
[\![\mathbf{0}]\!]_e &\stackrel{def}{=} \mathbf{0}\\
[\![\pi.P]\!]_e &\stackrel{def}{=} \pi.P\\
[\![P_1+P_2]\!]_e &\stackrel{def}{=} [\![P_1]\!]_e + [\![P_2]\!]_e\\
[\![\boldsymbol{\nu} x\, P]\!]_e &\stackrel{def}{=} \boldsymbol{\nu} x\, [\![P]\!]_e\\
[\![P_a \mid P_b]\!]_e &\stackrel{def}{=} \textstyle\sum_i M_i\alpha_i@d_i.(P_i \mid \sum_j N_j\beta_j@e_j.Q_j\{e_j{+}d_i/e_j\} + \sum_j N'_j t[m_j \dot{-} d_i].Q'_j)\\
&\quad + \textstyle\sum_j N_j\beta_j@e_j.(\sum_i M_i\alpha_i@d_i.P_i\{d_i{+}e_j/d_i\} + \sum_i M'_i t[n_i \dot{-} e_j].P'_i \mid Q_j)\\
&\quad + \textstyle\sum_i \sum_j M_i N_j[x=y]\tau.R_{ij}\\
&\quad + \textstyle\sum_i M'_i t[n_i].(P'_i \mid \sum_j N_j\beta_j@e_j.Q_j\{e_j{+}n_i/e_j\} + \sum_j N'_j t[m_j \dot{-} n_i].Q'_j)\\
&\quad + \textstyle\sum_j N'_j t[m_j].(\sum_i M_i\alpha_i@d_i.P_i\{d_i{+}m_j/d_i\} + \sum_i M'_i t[n_i \dot{-} m_j].P'_i \mid Q'_j)\\
[\![!\, P_a]\!]_e &\stackrel{def}{=} \textstyle\sum_i M_i\alpha_i@d_i.(P_i \mid !\,(\sum_j M_j\alpha_j@d_j.P_j\{d_j + d_i/d_j\} + \sum_j M'_j t[n_j \dot{-} d_i].P'_j))\\
&\quad + \textstyle\sum_i \sum_j M_i M_j[x=y]\tau.(R'_{ij} \mid !\, P_a)\\
&\quad + \textstyle\sum_i M'_i t[n_i].(P'_i \mid !\,(\sum_j M_j\alpha_j@d_j.P_j\{d_j + n_i/d_j\} + \sum_j M'_j t[n_j \dot{-} n_i].P'_j))
\end{aligned}
$$

図 3: 並行合成展開関数 $[\![\cdot]\!]_e$

定義 12 以下を満たす関係 $\mathcal{R}$ を時間抽象双模倣関係と呼ぶ.

$(P,Q) \in \mathcal{R}$ の時,

- $[\![P]\!]_a \stackrel{\alpha}{\rightharpoonup} P' \;\Rightarrow\; \exists Q'.\ [\![Q]\!]_a \stackrel{\alpha}{\rightharpoonup} Q' \wedge (P',Q') \in \mathcal{R}$
- $[\![Q]\!]_a \stackrel{\alpha}{\rightharpoonup} Q' \;\Rightarrow\; \exists P'.\ [\![P]\!]_a \stackrel{\alpha}{\rightharpoonup} P' \wedge (P',Q') \in \mathcal{R}$

□

$(P,Q) \in \mathcal{R}$ なる時間抽象双模倣関係 $\mathcal{R}$ が存在する時 $P \sim_T^{ta} Q$ と書く.

定義 13 $\sim_T^{ta} \;=\; \bigcup\{\mathcal{R} \mid \mathcal{R}$ は時間抽象双模倣関係 $\}$

□

時間待ちの長さが動的に決まる場合があるため，また複製演算子が存在するため，初めにプロセス式全体について時間経過動作を抽象することは不可能であり，入出力動作，内部動作，タイムアウト動作が実行されるたびに抽象化を行う．$P \sim_T^{ta} Q$ ならば，P と Q は時間経過動作を抽象化すると振舞いが等価であるとみなすことができる．

次に，入出力動作，内部動作，タイムアウト動作に着目し，時間経過動作には着目しない双模倣関係を定義する．

定義 14 以下を満たす対称な関係 $\mathcal{R}$ をタイムアウト双模倣関係と呼ぶ.

$(P,Q) \in \mathcal{R}$ の時,

- $P \stackrel{\alpha}{\rightharpoonup} P' \;\Rightarrow\; \exists Q'.\ Q \stackrel{\alpha}{\rightharpoonup} Q' \wedge (P',Q') \in \mathcal{R}$ (ただし $\alpha \neq \bullet$)
- $P \rightharpoonup^* \stackrel{\bullet}{\rightharpoonup} P' \;\Rightarrow\; \exists Q'.\ Q \rightharpoonup^* \stackrel{\bullet}{\rightharpoonup} P' \wedge (P',Q') \in \mathcal{R}$

□

$(P,Q) \in \mathcal{R}$ なるタイムアウト双模倣関係 $\mathcal{R}$ が存在する時 $P \sim_T^{to} Q$ と書く.

定義 15 $\sim_T^{to} \;=\; \bigcup\{\mathcal{R} \mid \mathcal{R}$ はタイムアウト双模倣関係 $\}$

□

この時，時間抽象双模倣なプロセスはタイムアウト双模倣であり，逆も成り立つ．時間経過動作に着目する必要がない性質についてプロセスの振舞いを解析する場合，構文的に時間経過動作を抽象してから 1 ステップずつプロセスの動作を調べればよい.

定理 5 $\sim_T^{ta} \;=\; \sim_T^{to}$ □

タイムアウト双模倣関係は入出力動作，内部動作および最も直近のタイムアウト動作のみに着目した時に双模倣関係が成り立つことを表している．定理 5 より，時間経過動作を抽象化した時に双模倣であることを表す時間抽象双模倣関係にある 2 つのプロセスは，タイムアウト双模倣関係でもあり，時間経過動作を抽象化して調べることができる.

5 おわりに

本稿では，時間付き π 計算のプロセスの時間動作をタイムアウト動作を残しながら抽象化する手法を提案した．時間付き π 計算は時間に関して細かい意味論を提供しているため，時間動作を詳細に調べることが可能である一方，状態を細かく分割することになるため状態爆発の問題が深刻になる．提案手法によりモデルの状態数を削減することができる．

提案した時間動作の抽象化手法は，展開定理に基づく並行プロセスから逐次プロセスへの変換，および時間経過動作の抽象からなる．時間経過動作の抽象では，タイムアウトから次のタイムアウトまでを一つの状態にまとめる．この時，時間動作の抽象化が行われたプロセスが双模倣であるならば，元になったプロセスは入出力動作，内部動作，タイムアウト動作のみに着目すれば双模倣であることを示した．つまり，時間動作に着目する必要のない振舞いを解析する時は，時間動作を抽象化すればよい．

今後の課題としては，提案手法が状態数の削減にどの程度の効果があるのか明らかにすることや，従来の π 計算に対する等価性判定を応用する方法を確立することなどが挙げられる．

参考文献

[1] Mads Dam. Model Checking Mobile Processes. *Information and Computation*, Vol. 129, No. 1, pp. 35–51, 1996.

[2] 桑原寛明, 結縁祥治, 阿草清滋. 時間付き π 計算によるリアルタイムオブジェクト指向言語の形式的記述. 情報処理学会論文誌, Vol. 45, No. 6, pp. 1498–1507, 2004.

[3] 桑原寛明, 結縁祥治, 阿草清滋. π 計算に対する時間拡張と合同的性質. 電子情報通信学会論文誌 D, Vol. J89-D, No. 4, pp. 632–641, 2006.

[4] Robin Milner. *Communication and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[5] Robin Milner, Joachim Parrow, and David Walker. A Calculus of Mobile Processes, Part I/II. *Information and Computation*, Vol. 100, pp. 1–77, 1992.

[6] Davide Sangiorgi. A Theory of Bisimulation for the π-Calculus. In *CONCUR'93*, Vol. 715 of *LNCS*, pp. 127–142. Springer, 1993.

[7] Davide Sangiorgi and David Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[8] 竹内泉. パイ計算による仕様を検証する論理体系. 情報処理学会論文誌：プログラミング, Vol. 46, No. SIG 11, pp. 57–65, 2005.

[9] Irek Ulidowski and Shoji Yuen. Extending Process Languages with Time. In *AMAST97*, Vol. 1394 of *LNCS*, pp. 524–538. Springer, 1997.

[10] Irek Ulidowski and Shoji Yuen. Process languages with discrete relative time based On the Ordered SOS format and rooted eager bisimulation. *The Journal of Logic and Algebraic Programming*, Vol. 60–61, pp. 401–460, 2004.

[11] Björn Victor and Faron Moller. The Mobility Workbench — A Tool for the π-Calculus. In *CAV'94: Computer Aided Verification*, Vol. 818 of *LNCS*, pp. 428–440. Springer, 1994.