π計算に対する時間拡張と合同的性質

桑原　寛明[†a)]　　結縁　祥治[†b)]　　阿草　清滋[†c)]

Congruence Properties for a Timed Extension of the π-Calculus

Hiroaki KUWABARA[†a)], Shoji YUEN[†b)], and Kiyoshi AGUSA[†c)]

あらまし　本論文では文献 [1] に基づくπ計算の時間拡張において，等価関係と擬順序関係が合同的性質をもつ十分条件を示す．時間経過が含まれるため，等価関係である時間拡張された双模倣関係は一般的には合同性を満たさない．しかし，チャネル名を束縛する入力プレフィックス以外のコンテクストに対しては合同であることを示す．ハードリアルタイム性をモデル化するために，遅延時間順関係と呼ぶ順序関係を定義する．遅延時間順関係は同じ入出力動作を一方のプロセスが他方より早いタイミングで実行できることを表す．遅延時間順関係は入力プレフィックス以外のコンテクストに対し合同であるが，並行合成されるプロセスは時間経過によって変換しないプロセスに制限される．ネットワーク構成が動的に変化するストリーミングシステムを用いて時間付きπ計算による記述と遅延時間順関係の例を示す．

キーワード　π計算，時間付きプロセス代数，合同性，実時間並行プログラム

1. まえがき

実時間システムには，デッドラインを守りながら動作することと適切なタイミングまで待機できることが要求される．一般に実時間システムのためのソフトウェアは時間制約をも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もちながら並行に動作するプログラムとして構成される．それぞれのコンポーネントの動作は他のコンポーネントの影響を受け，更に動作に時間制約が存在するため，このような実時間並行プログラムの振舞いを解析することは逐次プログラムに比べて難しい．そのため，実時間並行プログラムを正しく動作させるためには，システムの動作を詳細に解析する基本的手法が必要である．

実時間システムの振舞い解析を行うために，筆者らはシステムを実装するリアルタイムオブジェクト指向言語の意味記述を与えることを目的としてπ計算の時間拡張を提案した [1]．π計算 [2]〜[4] は高い表現能力をもつ並行システムの抽象計算モデルであり，代数的な性質に基づく様々な技法を適用することができる．代数的合同性によればシステムの一部を構成するコンポーネントを動作が等価なコンポーネントと交換してもシステム全体の性質は保たれる．π計算の意味論からシステム全体の信頼性の向上が期待できる．

本論文では，双模倣関係がチャネル名を束縛する入力プレフィックス以外のコンテクストに対して保存されることを示し，制限された環境ではモジュール性が保たれることを示す．合同性は，タイムアウトまでの時間が決められているタイムアウト動作がシステムの動作を変更しないこと，及びタイムアウト動作の時間変数に対する束縛関係が同一であることを保証する．

更に，合成演算子によって構築されたプロセスがデッドラインに関する仕様を満たすことを示すために，遅延時間順関係と呼ばれる関係を定義する．この関係は，並行合成できるプロセスは時間経過によって変化しないプロセスに制限すると，チャネル名を束縛する入力プレフィックス以外のコンテクストに対し合同であることが示される．ストリーミングシステムの例を用いて時間拡張について例示する．例で用いるシステムでは配信に時間制約が存在し，ネットワーク構成が動的に変化する．

本論文の構成は以下のとおりである．**2.** で時間付き

† 名古屋大学大学院情報科学研究科情報システム学専攻，名古屋市
Department of Information Engineering, Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8603 Japan
a) E-mail: kuwabara@agusa.i.is.nagoya-u.ac.jp
b) E-mail: yuen@is.nagoya-u.ac.jp
c) E-mail: agusa@is.nagoya-u.ac.jp

π 計算の構文と動作意味を定義する．**3.** で時間付き π 計算における双模倣関係の定義を示し，それらの関係が制限されたコンテクストに対して合同であることを示す．**4.** で遅延時間順関係について述べ，ストリーミングシステムの例を **5.** で示す．**6.** で関連研究に触れ，最後にまとめと今後の課題を述べる．

2. 時間付き π 計算

時間に関する性質を表現するために，文献 [1] に基づき従来の π 計算に離散時間の経過を表すプリミティブを導入する．

2.1 構　　文

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 $t[m]$ 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とする．$\mathcal{I} = \{\underline{0}, \underline{1}, \ldots\} \subset \mathcal{N}ame$，$\mathcal{N} = \mathcal{N}ame - \mathcal{I}$ とする．ここで $\underline{i}$ は自然数 i を表す名前である．$\tilde{x}$ は名前のリストを表し，i 番目の要素を x_i と書く．時間付き π 計算のプロセス全体の集合を $\mathcal{P}$ と書く．以下では，$l, x \in \mathcal{N}$，$n \in \mathcal{N}ame$，$m \in \mathcal{I}$ とする．n は $\mathcal{N}$ あるいは $\mathcal{I}$ の要素を表し，m は $\mathcal{I}$ の要素のみを表す．ここで $m \in \mathcal{I}$ に対し自然数 k が存在して m は $\underline{k}$ と書ける．$\underline{k}$ は k 単位時間を表し，$k > 0$ に対して $\underline{k-1}$ は $\underline{k}$ より一単位時間短い時間長を表す．$\tilde{y}$ の要素はすべて $\mathcal{N}$ に含まれ，$\tilde{w}$，$\tilde{z}$ の要素はすべて $\mathcal{N}ame$ に含まれるとする．

［定義 1］ 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．

$$\pi ::= x(\tilde{y}) \mid \overline{x}\langle\tilde{z}\rangle \mid \tau \mid t[n]$$
$$P ::= M \mid P_1 | P_2 \mid \boldsymbol{\nu} x\, P \mid !P$$
$$M ::= \mathbf{0} \mid \pi.P \mid M_1 + M_2$$ □

［定義 2］ 任意のプロセス式 P と $j \in \mathcal{I}$ に対し，構文

$$P_g ::= x(\tilde{y}).P \mid \overline{x}\langle\tilde{z}\rangle.P \mid \tau.P \mid t[j].P \mid P_g + P_g$$

によって定義できるプロセス P_g をガード付きプロセスと呼ぶ． □

Act を動作 π 及び $\overline{x}(\tilde{z})$ 全体の集合とする．$\tilde{w} \subseteq \tilde{z}$ であるとき $\overline{x}(\tilde{z})$ は $(\boldsymbol{\nu}\tilde{w})\overline{x}\langle\tilde{z}\rangle$ の略記である．新しい名前を含む $\tilde{z}$ を x を通して出力する動作を表す．

もしプロセス P のプレフィックスが $t[n]$ であり，かつ，$n \in \mathcal{N}$ ならば，P は次に入力，出力，τ などの動作も時間待ちも行うことはできない．時間経過動作の添字が $\mathcal{I}$ 以外の名前になる場合として動作不能プロセスを定義する．

［定義 3］ $P = t[l].P'$ において，l が入力動作によって束縛されていない場合 P は動作不能であるといい $P\!\uparrow$ と書く．また $P\!\uparrow$ のとき，

$$(x(\tilde{y}).P)\!\uparrow, (\overline{x}\langle\tilde{z}\rangle.P)\!\uparrow, (\tau.P)\!\uparrow, (t[n].P)\!\uparrow,$$
$$(P+Q)\!\uparrow, (P \mid Q)\!\uparrow, (\boldsymbol{\nu} z\, P)\!\uparrow, (!\,P)\!\uparrow$$

とする．ただしすべての i に対して $y_i \neq l$ である．動作不能でない場合，P は動作可能であるといい $P\!\downarrow$ と書く． □

$\uparrow$ は振舞いに関して保存されない．例えば，$P = \overline{x}\langle l\rangle.\mathbf{0} \mid x(n).t[n].P'$ とすると，$t[n].P'$ の n は入力動作 $x(n)$ によって束縛されているので P は動作可能である．τ 動作により $\mathbf{0} \mid t[l].P'$ に遷移すると $l \in \mathcal{N}$ であるため動作不能プロセスとなる．

［定義 4］ プロセス $x(\tilde{y}).P$ における名前 $\tilde{y}$ 及び $\boldsymbol{\nu} x\, P$ における名前 x のスコープは P に制限される．このとき，名前 $\tilde{y}$ 及び 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． □

$t[n]$ 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．$\mathcal{I}$ に含まれる名前に対する代入は不可能であるとする．

［定義 5］ 代入は $\mathcal{N}$ から $\mathcal{N}ame$ への関数である．プロセス P に代入 σ を適用したプロセス $P\sigma$ 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，$\sigma_{-\tilde{x}}$ は $\tilde{x}$ 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．

$$\mathbf{0}\sigma \stackrel{def}{=} \mathbf{0} \qquad (P \mid Q)\sigma \stackrel{def}{=} P\sigma \mid Q\sigma$$
$$(x(\tilde{y}).P)\sigma \stackrel{def}{=} x\sigma(\tilde{y}).P\sigma_{-\tilde{y}} \qquad (P+Q)\sigma \stackrel{def}{=} P\sigma + Q\sigma$$
$$(\overline{x}\langle\tilde{z}\rangle.P)\sigma \stackrel{def}{=} \overline{x\sigma}\langle\tilde{z\sigma}\rangle.P\sigma \qquad (\boldsymbol{\nu} x\, P)\sigma \stackrel{def}{=} \boldsymbol{\nu} x\, P\sigma_{-x}$$
$$(\tau.P)\sigma \stackrel{def}{=} \tau.P\sigma \qquad (!\,P)\sigma \stackrel{def}{=} !\,P\sigma$$
$$(t[n].P)\sigma \stackrel{def}{=} t[n\sigma].P\sigma$$

x が動作 $x(y)$ あるいは $\overline{x}\langle z\rangle$ のように最初の位置，すなわちサブジェクトとして出現する場合は $x\sigma \in \mathcal{N}$ でなければならないとする． □

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．$\overline{x}\langle\underline{5}\rangle.\mathbf{0}\mid x(n).t[n].P$ は τ 動作により $\mathbf{0}\mid t[\underline{5}].P$ に遷移するが，このとき n に $\underline{5}$ が代入される．

構造合同性を次のように定義する．

［定義 6］ 構造合同性は α 変換による関係 $\equiv_\alpha$(注1) と以下の公理を満たす最小の合同関係であり $\equiv$ と書く．

$$M_1+(M_2+M_3)\equiv(M_1+M_2)+M_3$$
$$M_1+M_2\equiv M_2+M_1$$
$$M_1+\mathbf{0}\equiv M_1$$
$$P_1\mid(P_2\mid P_3)\equiv(P_1\mid P_2)\mid P_3$$
$$P_1\mid P_2\equiv P_2\mid P_1$$
$$P_1\mid\mathbf{0}\equiv P_1$$
$$\boldsymbol{\nu}z\,\boldsymbol{\nu}w\,P\equiv\boldsymbol{\nu}w\,\boldsymbol{\nu}z\,P$$
$$\boldsymbol{\nu}z\,\mathbf{0}\equiv\mathbf{0}$$
$$\boldsymbol{\nu}z\,(P\mid Q)\equiv P\mid\boldsymbol{\nu}z\,Q\qquad z\notin\mathsf{fn}(P)$$
$$!\,P\equiv P\mid!\,P$$
□

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 として振る舞う．タイムアウトは $a.P+t[\underline{5}].Q$ のように時間経過動作と非決定選択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 単位時間経過したときにタイムアウトし Q へ遷移する．

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）$x(\widetilde{y})$：x を通して $\widetilde{y}$ を受信
- （出力）$\overline{x}\langle\widetilde{z}\rangle$：$x$ を通して $\widetilde{z}$ を送信
- （τ 動作）τ：外部からは不可視な内部アクション
- （選択）$P+Q$：プロセス P，Q のうち実行可能なプロセスを選択し実行する
- （並行）$P|Q$：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 ｜ 演算子によって結合しているとみなす

2.2 ラベル付き遷移による動作意味

$\mathcal{P}$ 上の遷移関係 $\{\xrightarrow{\alpha}\mid\alpha\in Act\cup\{\bullet\}\}\cup\{\rightarrow\}$ 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．

$$\text{OUT}:\ \frac{}{\overline{x}\langle\widetilde{z}\rangle.P\xrightarrow{\overline{x}\langle\widetilde{z}\rangle}P}\qquad \text{INPUT}:\ \frac{}{x(\widetilde{y}).P\xrightarrow{x(\widetilde{z})}P\{\widetilde{z}/\widetilde{y}\}}$$
$$\text{TAU}:\ \frac{}{\tau.P\xrightarrow{\tau}P}\qquad \text{TIMEOUT}:\ \frac{}{t[\underline{0}].P\xrightarrow{\bullet}P}$$
$$\text{T-SUM-L}:\ \frac{P\xrightarrow{\bullet}P'}{P+Q\xrightarrow{\bullet}P'}\qquad \text{SUM-L}:\ \frac{P\xrightarrow{\alpha}P'\quad Q\not\xrightarrow{\bullet}}{P+Q\xrightarrow{\alpha}P'}\ \alpha\neq\bullet$$
$$\text{COMM-L}:\ \frac{P\xrightarrow{\overline{x}(\widetilde{z})}P'\quad Q\xrightarrow{x(\widetilde{z})}Q'}{P\mid Q\xrightarrow{\tau}P'\mid Q'}$$
$$\text{PAR-L}:\ \frac{P\xrightarrow{\alpha}P'}{P\mid Q\xrightarrow{\alpha}P'\mid Q}\ \widetilde{w}\cap\mathsf{fn}(Q)=\emptyset\ \text{if}\ \alpha=(\boldsymbol{\nu}\widetilde{w})\,\overline{x}\langle\widetilde{z}\rangle$$
$$\text{CLOSE-L}:\ \frac{P\xrightarrow{\overline{x}(\widetilde{z})}P'\quad Q\xrightarrow{x(\widetilde{z})}Q'}{P\mid Q\xrightarrow{\tau}\boldsymbol{\nu}\widetilde{w}\,(P'\mid Q')}\ \widetilde{w}\notin\mathsf{fn}(Q),\widetilde{w}\subseteq\widetilde{z}$$
$$\text{RES}:\ \frac{P\xrightarrow{\alpha}P'}{\boldsymbol{\nu}\widetilde{x}\,P\xrightarrow{\alpha}\boldsymbol{\nu}\widetilde{x}\,P'}\ \mathsf{n}(\alpha)\cap\widetilde{x}=\emptyset$$
$$\text{OPEN}:\ \frac{P\xrightarrow{\overline{x}(\widetilde{z})}P'}{\boldsymbol{\nu}\widetilde{w}\,P\xrightarrow{\overline{x}(\widetilde{z})}P'}\ \widetilde{w}\subseteq\widetilde{z},x\notin\widetilde{w}$$
$$\text{REP-ACT}:\ \frac{P\xrightarrow{\alpha}P'}{!P\xrightarrow{\alpha}P'\mid!P}$$
$$\text{REP-COMM}:\ \frac{P\xrightarrow{\overline{x}(\widetilde{z})}P'\quad P\xrightarrow{x(\widetilde{z})}P''}{!P\xrightarrow{\tau}(P'\mid P'')\mid!P}$$
$$\text{REP-CLOSE}:\ \frac{P\xrightarrow{\overline{x}(\widetilde{z})}P'\quad P\xrightarrow{x(\widetilde{z})}P''}{!P\xrightarrow{\tau}(\boldsymbol{\nu}\widetilde{w}\,(P'\mid P''))\mid!P}\ \widetilde{w}\notin\mathsf{fn}(P),\widetilde{w}\subseteq\widetilde{z}$$

図 1 遷移規則
Fig. 1 The rules for actions.

$$\text{PASS}_\text{T}:\ \frac{}{t[\underline{k}].P\rightarrow t[\underline{k-1}].P}\ \text{if}\ k>0\qquad \text{INACT}_\text{T}:\ \frac{}{\mathbf{0}\rightarrow\mathbf{0}}$$
$$\text{OUT}_\text{T}:\ \frac{}{\overline{x}\langle\widetilde{z}\rangle.P\rightarrow\overline{x}\langle\widetilde{z}\rangle.P}\qquad \text{IN}_\text{T}:\ \frac{}{x(\widetilde{y}).P\rightarrow x(\widetilde{y}).P}$$
$$\text{SUM}_\text{T}:\ \frac{P\rightarrow P'\quad Q\rightarrow Q'}{P+Q\rightarrow P'+Q'}$$
$$\text{PAR}_\text{T}:\ \frac{P\rightarrow P'\quad Q\rightarrow Q'}{P\mid Q\rightarrow P'\mid Q'}\ \text{if}\ P\mid Q\not\xrightarrow{\tau}$$
$$\text{RES}_\text{T}:\ \frac{P\rightarrow P'}{\boldsymbol{\nu}\widetilde{x}\,P\rightarrow\boldsymbol{\nu}\widetilde{x}\,P'}\qquad \text{REP}_\text{T}:\ \frac{P\rightarrow P'}{!P\rightarrow!P'}\ \text{if}\ P\mid P\not\xrightarrow{\tau}$$
$$\text{STRUCT}_\text{T}:\ \frac{P\rightarrow P'}{Q\rightarrow Q'}\ \text{if}\ P\equiv Q,\ P'\equiv Q'$$

図 2 時間経過規則
Fig. 2 The rules for time passing.

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，COMM-L 規則，PAR-L 規則，CLOSE-L 規則においてそれぞれ P と Q を入れ換えた T-SUM-R 規則，SUM-R 規則，COMM-R 規則，PAR-R 規則，CLOSE-R 規則が省略されている．PAR-L 規則，PAR-R 規則，RES 規則，REP-ACT 規則においては $\alpha=\bullet$ の場合も含む．

(注1)：P に対し α 変換のみを適用することによって Q が得られるならば $P\equiv_\alpha Q$.

$P \xrightarrow{\bullet} P'$ はタイムアウトによって P から P' に遷移することを表す．タイムアウトによる遷移は選択演算子においてのみ意味をもち，タイムアウトが発生したプロセスは入出力動作を行う他のプロセスよりも優先して選択される．$t[0].P + t[0].Q$ のようにタイムアウトするプロセスが複数存在する場合は非決定的にいずれかのプロセスが選択される．$t[3].\overline{x}.\mathbf{0} + t[5].y.\mathbf{0}$ のようにタイムアウトまでの時間が異なるプロセスを選択しようとしても，$t[3].\overline{x}.\mathbf{0}$ が $t[5].y.\mathbf{0}$ よりも先にタイムアウトするため必ず $t[3].\overline{x}.\mathbf{0}$ が選択される．3 単位時間後に $\overline{x}$ が実行可能になり，5 単位時間後に y が実行可能になるプロセスは $t[3].(\overline{x}.\mathbf{0} + t[2].(\overline{x}.\mathbf{0} + y.\mathbf{0}))$ のように記述する．

$P \xrightarrow{\alpha} P'$（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，$P \rightharpoonup P'$ 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．

3. 時間付き双模倣関係の合同性

本章では，文献 [1] に基づいて時間付き双模倣関係を示し，合同性が成り立つ十分条件を示す．

3.1 時間付き双模倣関係

時間付き π 計算におけるプロセス間の双模倣関係を定義する．時間付き双模倣関係は直観的には以下の条件を満たす関係である．

- 双方のプロセスの入出力動作の列が同じ
- 双方のプロセスにおいて時間待ちが発生するタイミングと時間待ちの長さが同じ

[定義 7] $\alpha \in Act$ に対して強遷移関係を以下のように定義する．

- $\Rightarrow_\alpha \stackrel{def}{=} \xrightarrow{\bullet}{}^* \Rightarrow_\alpha \Rightarrow \xrightarrow{\bullet}{}^*$
- $\rightsquigarrow \stackrel{def}{=} \xrightarrow{\bullet}{}^* \rightharpoonup \xrightarrow{\bullet}{}^*$ □

[定義 8] 以下を満たす対称な関係 $\mathcal{R}$ を時間付き強双模倣関係と呼び，最大の関係を $\sim_{\mathcal{T}}$ と書く．

$(P, Q) \in \mathcal{R}$ のとき，

- $P\!\uparrow \;\Rightarrow\; Q\!\uparrow$
- $P \xrightarrow{\alpha} P' \;\Rightarrow\; \exists Q'.\, Q \xrightarrow{\alpha} Q' \wedge (P', Q') \in \mathcal{R}$
- $P \rightsquigarrow P'' \;\Rightarrow\; \exists Q''.\, Q \rightsquigarrow Q'' \wedge (P'', Q'') \in \mathcal{R}$

□

時間付き強双模倣関係はプロセスの振舞等価性の基礎である．もしプロセス P と Q が時間付き強双模倣関係にあれば，実行系列から P と Q を区別することは不可能である．時間付き強双模倣関係では，入出力動作，内部動作，時間経過動作を等しく扱う．例えば，プロセス $t[1].\overline{x}.\mathbf{0} \mid t[1].y.\mathbf{0}$ と $t[1].(\overline{x}.y.\mathbf{0} + y.\overline{x}.\mathbf{0})$ は，以下に示す関係 $\mathcal{R}$ から時間付き強双模倣関係であることが分かる．

$$\begin{aligned}\mathcal{R} = \{&(t[1].\overline{x}.\mathbf{0} \mid t[1].y.\mathbf{0},\ t[1].(\overline{x}.y.\mathbf{0} + y.\overline{x}.\mathbf{0})),\\ &(\overline{x}.\mathbf{0} \mid y.\mathbf{0},\ \overline{x}.y.\mathbf{0} + y.\overline{x}.\mathbf{0}),\\ &(\mathbf{0} \mid y.\mathbf{0},\ y.\mathbf{0}),\\ &(\overline{x}.\mathbf{0} \mid \mathbf{0},\ \overline{x}.\mathbf{0}),\\ &(\mathbf{0} \mid \mathbf{0},\ \mathbf{0})\}\end{aligned}$$

次に τ 動作を抽象した時間付き弱双模倣関係を定義する．

[定義 9] $\alpha \in Act$ に対して弱遷移関係を以下のように定義する．

- $\longrightarrow_\tau \stackrel{def}{=} \xrightarrow{\tau}{}^*$
- $\xrightarrow{\alpha}_\tau \stackrel{def}{=} \longrightarrow_\tau \xrightarrow{\alpha} \longrightarrow_\tau$
- $\rightsquigarrow_\tau \stackrel{def}{=} \longrightarrow_\tau \rightsquigarrow \longrightarrow_\tau$ □

以下では簡単のために，$\xrightarrow{\widehat{\tau}}_\tau$ は $\longrightarrow_\tau$ を，τ 以外の動作 β に対し $\xrightarrow{\widehat{\beta}}_\tau$ は $\xrightarrow{\beta}_\tau$ を表す記法を用いる．

[定義 10] 以下を満たす対称な関係 $\mathcal{R}$ を時間付き弱双模倣関係と呼び，最大の関係を $\approx_{\mathcal{T}}$ と書く．

$(P, Q) \in \mathcal{R}$ のとき，

- $P\!\uparrow \;\Rightarrow\; Q\!\uparrow$
- $P \xrightarrow{\alpha} P' \;\Rightarrow\; \exists Q'.\, Q \xrightarrow{\widehat{\alpha}}_\tau Q' \wedge (P', Q') \in \mathcal{R}$
- $P \rightsquigarrow P'' \;\Rightarrow\; \exists Q''.\, Q \rightsquigarrow_\tau Q'' \wedge (P'', Q'') \in \mathcal{R}$

□

$\sim_{\mathcal{T}}$ 及び $\approx_{\mathcal{T}}$ は等価関係である．

3.2 時間付き双模倣関係の合同性

$\sim_{\mathcal{T}}$ 及び $\approx_{\mathcal{T}}$ が制限された入力動作以外の演算子について保存されることを示す．

[定義 11] 時間経過動作の添字以外の名前を束縛する入力プレフィックスを除くコンテクストを以下のように再帰的に定義する．

$$\begin{aligned}C_{nit}[\cdot] ::= {} & [\cdot] \mid \overline{x}\langle\widetilde{z}\rangle.C_{nit}[\cdot] \mid x(\widetilde{y}).C_{nit}[\cdot]\\ & \mid \tau.C_{nit}[\cdot] \mid t[n].C_{nit}[\cdot] \mid \tau.C_{nit}[\cdot] + R\\ & \mid \overline{x}\langle\widetilde{z}\rangle.C_{nit}[\cdot] + R \mid x(\widetilde{y}).C_{nit}[\cdot] + R\\ & \mid C_{nit}[\cdot] | S \mid \boldsymbol{\nu} x\, C_{nit}[\cdot] \mid !C_{nit}[\cdot]\end{aligned}$$

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．また $\widetilde{y}$ はコンテクストを適用するプロセスに出現する自由な時間経過動作の添字のみを含むとす

る． □

[定理 1] $P \sim_{\mathcal{T}} Q$ ならば $C_{nit}[P] \sim_{\mathcal{T}} C_{nit}[Q]$ である．

(証明) コンテクストの構造に関する帰納法により証明する．ここでは $C[\cdot] = t[n].C'[\cdot]$ の場合について示す．もし $n \in \mathcal{N}$ ならば $C[P]\uparrow$ かつ $C[Q]\uparrow$ である．$n \in \mathcal{I}$ ならば $n = \underline{i}$ かつ $i > 0$ なる自然数 i が存在し，任意のアクション α に対し $C[P] \stackrel{\alpha}{\nrightarrow}$ であるので，$t[\underline{i}].C'[P] \sim_{\mathcal{T}} t[\underline{i}].C'[Q]$ を示せばよい．$i = 1$ とすると，$t[\underline{1}].C'[P] \rightsquigarrow C'[P]$ かつ $t[\underline{1}].C'[Q] \rightsquigarrow C'[Q]$ である．コンテクストに関する帰納法の仮定より $C'[P] \sim_{\mathcal{T}} C'[Q]$ なので $t[\underline{1}].C'[P] \sim_{\mathcal{T}} t[\underline{1}].C'[Q]$ である．次に，$t[\underline{i}].C'[P] \rightsquigarrow t[\underline{i-1}].C'[P]$ かつ $t[\underline{i}].C'[Q] \rightsquigarrow t[\underline{i-1}].C'[Q]$ であり，帰納法の仮定より $t[\underline{i-1}].C'[P] \sim_{\mathcal{T}} t[\underline{i-1}].C'[Q]$ なので $t[\underline{i}].C'[P] \sim_{\mathcal{T}} t[\underline{i}].C'[Q]$ である． □

上記の定義において動作可能性は以下のような意味をもつ．$v \in \mathcal{N}$，$Q_1 \not\downarrow$，$Q_2 \not\downarrow$，$P_1 = \tau.Q_1 + t[v].Q_2$，$P_2 = \tau.Q_1$ とする．このとき，$P_1\uparrow$，$P_2 \not\downarrow$ ゆえ $P_1 \not\approx_{\mathcal{T}} P_2$ である．コンテクスト $\overline{a}\langle\underline{0}\rangle.\mathbf{0} \,|\, a(v).[\cdot]$ を適用して $P_1' = \overline{a}\langle\underline{0}\rangle.\mathbf{0} \,|\, a(v).P_1$，$P_2' = \overline{a}\langle\underline{0}\rangle.\mathbf{0} \,|\, a(v).P_2$ とすると，$P_1' \not\downarrow$，$P_2' \not\downarrow$ であり，$P_1' \xrightarrow{\tau} Q_2$ に対して $P_2' \xrightarrow{\tau} \tau.Q_1$ となるので $P_1' \not\sim_{\mathcal{T}} P_2'$ である．ここで時間付き強双模倣関係の一つ目の条件がない場合を考えると，P_1 と P_2 は時間付き強双模倣になるが，P_1' と P_2' は時間付き強双模倣ではなく合同性が成り立たない．

[定理 2] $P \approx_{\mathcal{T}} Q$ ならば $C_{nit}[P] \approx_{\mathcal{T}} C_{nit}[Q]$ である．

(証明) コンテクストの構造に関する帰納法による． □

時間付き双模倣関係は従来の π 計算と同様に通信チャネルとして利用される名前に対する代入に対しては保存されない．文献 [4] と同様に選択演算子そのものに対しては合同性は保存されない．例えば，$t[\underline{0}].P \approx_{\mathcal{T}} P$ であるが，$t[\underline{0}].P + Q \not\approx_{\mathcal{T}} P + Q$ である．しかし，これは選択演算子の左辺についてガードされたコンテクストに埋め込まれていない．ガードされた選択については，例えば，$\overline{a}.t[\underline{0}].P + \overline{b}.Q \approx_{\mathcal{T}} \overline{a}.P + \overline{b}.Q$ である．このことにより時間付き弱双模倣関係は選択演算に対し保存される(注2)．

4. 遅延時間順関係

本章では，遅延時間の長さに着目し，同じ入出力動作を行う二つのプロセスにおいて動作を実行するタイミングがいずれのプロセスが他方より早いかあるいは遅いかを表す順序関係を与える．

[定義 12] 以下の条件を満たす関係 $\mathcal{R}$ を遅延時間順関係と呼び，最大の関係を $\precsim_{\mathcal{T}}$ と書く．

$(P, Q) \in \mathcal{R}$ のとき，

$$
\begin{aligned}
&P\uparrow \Rightarrow Q\uparrow \\
&Q\uparrow \Rightarrow P\uparrow \\
&P \xrightarrow{\alpha} P' \Rightarrow \exists Q'.\ Q \rightsquigarrow^* \xrightarrow{\alpha} Q' \wedge (P', Q') \in \mathcal{R} \\
&P \rightsquigarrow P' \Rightarrow \exists Q'.\ Q \rightsquigarrow Q' \wedge (P', Q') \in \mathcal{R} \\
&Q \xrightarrow{\alpha} Q' \Rightarrow \exists P'.\ P \xrightarrow{\alpha} P' \wedge (P', Q') \in \mathcal{R} \\
&Q \rightsquigarrow Q' \Rightarrow (P, Q') \in \mathcal{R} \vee \\
&\qquad\qquad (\exists P'.\ P \rightsquigarrow P' \wedge (P', Q') \in \mathcal{R})
\end{aligned}
$$

□

直観的には，$P \precsim_{\mathcal{T}} Q$ ならば時間経過動作による遷移の回数は P より Q の方が多いが，その他の動作による遷移は P と Q のいずれも同じである．もし P が時間経過以外の何らかの動作が実行可能ならば，Q は 0 単位時間以上後に同じ動作が実行できる．P が時間経過遷移するのであれば，時間経過遷移の回数は P の方が少ないので Q は P と同様に時間経過遷移できなければならない．同じ理由から Q が時間経過以外の動作を実行できるならば，P は同じ動作が即座に実行できなければならない．Q が時間経過遷移するとき，P は遷移後に関係が保たれるのであれば時間経過遷移することができる．明らかに $\precsim_{\mathcal{T}}$ は擬順序である．

遅延時間順関係は，並行合成されるプロセスには制限があるが，時間経過動作の添字以外の名前を束縛する入力プレフィックスと複製を除くすべての演算子について保存される．

[定義 13] $P \stackrel{\tau}{\nrightarrow}$ かつ $P \rightsquigarrow P$ のとき，P は時間安定であり，$TS(P)$ と書く．$derive(P) = \{P' \,|\, P(\stackrel{\alpha}{\rightarrow})^* P'\}$ とする．このとき，$\forall Q \in derive(P).\, TS(Q)$ ならば，P は時間独立であり，$TI(P)$ と書く． □

$TI(P)$ の定義より明らかに $TI(P)$ ならば $P \xrightarrow{\alpha} P'$ なるすべての P' に対して $TI(P')$ である．

[定義 14] 並行合成するプロセスが時間独立なプロセスに制限されるコンテクスト C_{ti} を以下のように定義する．

$$C_{ti}[\cdot] ::= [\cdot] \mid \overline{x}\langle\widetilde{z}\rangle.C_{ti}[\cdot] \mid x(\widetilde{y}).C_{ti}[\cdot]$$

(注2)：強双模倣関係についても同様．

$$
\begin{aligned}
&\mid \tau.C_{ti}[\cdot] \mid t[n].C_{ti}[\cdot] \mid \tau.C_{ti}[\cdot] + R\\
&\mid \overline{x}\langle\widetilde{z}\rangle.C_{ti}[\cdot] + R \mid x(\widetilde{y}).C_{ti}[\cdot] + R\\
&\mid C_{ti}[\cdot] | S \mid \boldsymbol{\nu} x\, C_{ti}[\cdot]
\end{aligned}
$$

ここで R はガード付きプロセス，S は時間独立なプロセスであるとする．また $\widetilde{y}$ はコンテクストを適用するプロセスに出現する自由な時間経過動作の添字のみを含むとする． □

[定理 3] $P \precsim_{\mathcal{T}} Q$ ならば $C_{ti}[P] \precsim_{\mathcal{T}} C_{ti}[Q]$ である．

(証明) コンテクストの構造に関する帰納法によって証明する．

[基底段階]

$C_{ti} = [\cdot]$ の場合，$C_{ti}[P] = P$，$C_{ti}[Q] = Q$ であるので，仮定より明らかに $C_{ti}[P] \precsim_{\mathcal{T}} C_{ti}[Q]$ である．

[帰納段階]

ここでは $C_{ti} = C'_{ti} | S$ の場合を示す．$\mathcal{R} = \{(C'_{ti}[P] \mid S, C'_{ti}[Q] \mid S) \mid P \precsim_{\mathcal{T}} Q, TI(S)\}$ とする．S は時間独立なプロセスであるため $S \not\gamma$ である．同じ理由により $S \rightsquigarrow S$ であるので $S \rightsquigarrow^* S$ となる．

- $C_{ti}[P]\uparrow$ の場合，$S \not\gamma$ なので $C'_{ti}[P]\uparrow$ である．ここで帰納法の仮定より $C'_{ti}[P] \precsim_{\mathcal{T}} C'_{ti}[Q]$ ゆえ $C'_{ti}[Q]\uparrow$ であるので，定義 3 より $C_{ti}[Q]\uparrow$ となる．逆も同様である．
- $C'_{ti}[P] \mid S \xrightarrow{\alpha}$ の場合，

Case1 $\alpha \neq \tau$ のとき

Case1.1 $C'_{ti}[P] \mid S \xrightarrow{\alpha} P' \mid S$（ただし $C'_{ti}[P] \xrightarrow{\alpha} P'$）の場合，帰納法の仮定より $C'_{ti}[P] \precsim_{\mathcal{T}} C'_{ti}[Q]$ であるので P' に対し $C'_{ti}[Q] \rightsquigarrow^* \xrightarrow{\alpha} Q'$ かつ $P' \precsim_{\mathcal{T}} Q'$ なる Q' が存在する．$C'_{ti}[Q] \rightsquigarrow^* Q^*$ とすると，S は時間独立であるので $C'_{ti}[Q] \mid S \rightsquigarrow^* Q^* \mid S$ となる．更に $Q^* \mid S \xrightarrow{\alpha} Q' \mid S$ とできて，$P' \precsim_{\mathcal{T}} Q'$ ゆえ $(P' \mid S, Q' \mid S) \in \mathcal{R}$ である．

Case1.2 $C'_{ti}[P] \mid S \xrightarrow{\alpha} C'_{ti}[P] \mid S'$（ただし $S \xrightarrow{\alpha} S'$）の場合，$C'_{ti}[Q] \mid S \xrightarrow{\alpha} C'_{ti}[Q] \mid S'$ とできる．ここで，S は時間独立であるため S' も時間独立であり，よって $(C'_{ti}[P] \mid S', C'_{ti}[Q] \mid S') \in \mathcal{R}$ である．

Case2 $\alpha = \tau$ のとき

Case2.1 $C'_{ti}[P] \mid S \xrightarrow{\tau} P' \mid S$（ただし $C'_{ti}[P] \xrightarrow{\tau} P'$）の場合，Case1.1 と同様．

Case2.2 $C'_{ti}[P] \mid S \xrightarrow{\tau} P' \mid S'$（ただし $C'_{ti}[P] \xrightarrow{\alpha} P'$，$S \xrightarrow{\overline{\alpha}} S'$）の場合，帰納法の仮定より $C'_{ti}[P] \precsim_{\mathcal{T}} C'_{ti}[Q]$ であるので P' に対し $C'_{ti}[Q] \rightsquigarrow^* \xrightarrow{\alpha} Q'$ かつ $P' \precsim_{\mathcal{T}} Q'$ なる Q' が存在する．$C'_{ti}[Q] \rightsquigarrow^* Q^*$ とすると，S は時間独立であるので $C'_{ti}[Q] \mid S \rightsquigarrow^* Q^* \mid S$ となる．ここで $Q^* \xrightarrow{\alpha} Q'$ かつ $S \xrightarrow{\overline{\alpha}} S'$ ゆえ，$Q^* \mid S \xrightarrow{\tau} Q' \mid S'$ とできて，S' も時間独立であるため $(P' \mid S', Q' \mid S') \in \mathcal{R}$ である．

- $C'_{ti}[Q] \mid S \rightsquigarrow Q'' \mid S$（ただし $C'_{ti}[Q] \rightsquigarrow Q''$）の場合，帰納法の仮定から $C'_{ti}[P] \precsim_{\mathcal{T}} Q''$ あるいは Q'' に対し $C'_{ti}[P] \rightsquigarrow P''$ かつ $P'' \precsim_{\mathcal{T}} Q''$ なる P'' が存在する．$C'_{ti}[P] \precsim_{\mathcal{T}} Q''$ ならば $(C'_{ti}[P] \mid S, Q'' \mid S) \in \mathcal{R}$ である．そうでなければ $S \rightsquigarrow S$ ゆえ $C'_{ti}[P] \mid S \rightsquigarrow P'' \mid S$ とできて，$P'' \precsim_{\mathcal{T}} Q''$ であるので $(P'' \mid S, Q'' \mid S) \in \mathcal{R}$ である．
- $C'_{ti}[P] \mid S \rightsquigarrow P'' \mid S$，$C'_{ti}[Q] \mid S \xrightarrow{\alpha} Q' \mid S$，$C'_{ti}[Q] \mid S \xrightarrow{\tau} Q' \mid S'$，$C'_{ti}[Q] \mid S \xrightarrow{\alpha} C'_{ti}[Q] \mid S'$ の場合についても同様に示すことができる．

以上より $\mathcal{R} \subseteq \precsim_{\mathcal{T}}$ である． □

遅延時間順関係は時間経過動作を含むプロセスとの並行合成において保存されない．例えば，$P = a.\mathbf{0}$，$Q = t[1].a.\mathbf{0}$，$R = t[2].b.\mathbf{0}$ としたとき，$P \precsim_{\mathcal{T}} Q$ である．しかし，$P \mid R \xrightarrow{a} \mathbf{0} \mid R$ に対し $Q \mid R \rightsquigarrow \xrightarrow{a} \mathbf{0} \mid t[1].b.\mathbf{0}$ であるため $P \mid R \not\precsim_{\mathcal{T}} Q \mid R$ となり関係は成り立たない．$P \precsim_{\mathcal{T}} Q$ のとき $Q \mid R$ の方が Q の次の入出力あるいは τ 動作までに多くの時間経過動作を必要とする．R が時間経過動作をプレフィックスにもつならば，R の次の入出力動作までに必要な時間経過は逆に少なくなる．遅延時間順関係はプロセスが遷移しても常に保たれなければならないため，並行合成されるプロセスが時間経過動作を含むことはできない．

$P \precsim_{\mathcal{T}} Q$ ならば P と Q は時間経過動作以外の動作について同じ実行系列をもっている．これらの実行系列は外部から観測不可能な τ 動作も含んでいる．次に，τ 動作以外の外部から観測可能な動作のみに着目した遅延時間順関係を与える．

[定義 15] 以下の条件を満たす関係 $\mathcal{R}$ を弱遅延時間順関係と呼び，最大の関係を $\lesssim_{\mathcal{T}}$ と書く．

$$
\begin{aligned}
&P\uparrow \Rightarrow Q\uparrow\\
&Q\uparrow \Rightarrow P\uparrow\\
&P \xrightarrow{\alpha} P' \Rightarrow \exists Q'.\ Q \rightsquigarrow_\tau^* \xrightarrow{\widehat{\alpha}}_\tau Q' \wedge (P', Q') \in \mathcal{R}\\
&P \rightsquigarrow P' \Rightarrow \exists Q'.\ Q \rightsquigarrow_\tau Q' \wedge (P', Q') \in \mathcal{R}\\
&Q \xrightarrow{\alpha} Q' \Rightarrow \exists P'.\ P \xrightarrow{\widehat{\alpha}}_\tau P' \wedge (P', Q') \in \mathcal{R}\\
&Q \rightsquigarrow Q' \Rightarrow P \gtrsim_{\mathcal{T}} Q' \vee\\
&\qquad\qquad (\exists P'.\ P \rightsquigarrow_\tau P' \wedge (P', Q') \in \mathcal{R})
\end{aligned}
$$

□

$\lesssim_{\mathcal{T}}$ は擬順序である．

［定理 4］ $P \gtrapprox_{\mathcal{T}} Q$ ならば $C_{ti}[P] \gtrapprox_{\mathcal{T}} C_{ti}[Q]$ である．（証明）定理 3 と同様にコンテクストの構造に関する帰納法による． □

ここで，並行合成の場合には，合成されるプロセスは時間独立であるため τ 遷移しない．そのため，τ 遷移が抽象されているが定理 3 の場合と同様にして示すことができる．

弱遅延時間順関係も遅延時間順関係と同様に入力プレフィックスに対して保存されない．$P \gtrapprox_{\mathcal{T}} Q$ であることを示すためには，P 及び Q から入力以外のプレフィックスと並行合成された時間独立なプロセスを取り除いた P' と Q' について $P' \gtrapprox_{\mathcal{T}} Q'$ であることを示せばよい．

プロセス P と Q が弱遅延時間順関係にあるとき，入出力動作について P と Q は双模倣であり，ある入出力動作から次の入出力動作までの間に経過する時間の長さは P の方が常に短い．このことから，任意の実行系列において任意の二つの入出力動作の間で経過する時間は必ず P の方が短いことがいえる．実時間システムにおける時間制約の一つにデッドラインがあるが，実時間システムはデッドラインの到達よりも早く動作を行う必要がある．システムの実装を P，デッドラインの仕様を Q とすると $P \gtrapprox_{\mathcal{T}} Q$ を示すことでシステムがデッドラインの仕様よりも早い，つまりデッドラインを破らないことが示せる．

5. 例

本章ではストリーミング配信システムの簡単な例を示す．システムは映像を配信するサーバ，配信される映像を受信して再生するプレイヤ，サーバとプレイヤの間の通信路を確保する二つのルータからなる．プレイヤはいずれかのルータに接続して配信を受けるが，もし一定時間配信されてこなければもう一方のルータに接続し直す．ルータに接続してから配信の開始まで一定時間必要であり，ここで必要な時間はルータによって異なるとする．またサーバから送信される映像は圧縮されており，プレイヤで再生する前に解凍しなければならないとする．

サーバ $Server$ は以下のように記述できる．

$$Server \stackrel{def}{=} !\,\overline{v}\langle video\rangle$$

サーバは名前 v を通して映像を配信する．名前 $video$ は映像の 1 フレームを表す．実際にはフレームは次々と切り換わっていくが，簡単のためにそれらはすべて同じ名前 $video$ で表す．

二つのルータ $Router_i$ は

$$\begin{aligned} Router_1 \stackrel{def}{=} {} & !\,rel_1.attach(l).(\boldsymbol{\nu} u, g) \\ & (\overline{l}\langle \underline{2}, u, rel_1\rangle.v(video).\overline{u}\langle video\rangle.\overline{g} \\ & \mid !\,(g.v(video).\overline{u}\langle video\rangle.\overline{g} + g)) \\ Router_2 \stackrel{def}{=} {} & !\,rel_2.attach(l).(\boldsymbol{\nu} u, g) \\ & (\overline{l}\langle \underline{4}, u, rel_2\rangle.v(video).\overline{u}\langle video\rangle.\overline{g} \\ & \mid !\,(g.v(video).\overline{u}\langle video\rangle.\overline{g} + g)) \end{aligned}$$

と記述できる．ルータは名前 $attach$ によりプレイヤから接続要求を受け，配信開始までの時間，映像を送るための名前 u，プレイヤとの接続を切断する rel_i をプレイヤに送信する．その後，名前 v によりサーバから受信した映像を名前 u を通してプレイヤに送信する．もしサーバからの受信に失敗したならばプレイヤからの切断を待つ．この例では配信開始までの時間を $Router_1$ で 2 単位時間，$Router_2$ で 4 単位時間とする．

プレイヤ $Player$ の記述を以下に示す．

$$\begin{aligned} & Player \\ & \stackrel{def}{=} (\boldsymbol{\nu} l, g, h)\,(\overline{attach}\langle l\rangle.\overline{h} \mid !h.l(n, v, r).t[n].\overline{g}\langle v, r\rangle \\ & \quad \mid !\,g(v, r).(v(video).t[\underline{1}].\overline{play}\langle video\rangle.\overline{g}\langle v, r\rangle \\ & \quad + t[\underline{1}].\overline{attach}\langle l\rangle.\overline{r}.\overline{h})) \end{aligned}$$

プレイヤははじめに名前 $attach$ によりいずれかのルータに接続要求を出し，名前 l を通して配信開始までの時間，映像を受けるための名前 v，ルータとの接続を切断する r を受信する．その後，受信した時間だけ待機する．次に名前 v によりルータからの映像の配信を待機する．配信を受けたら 1 単位時間で圧縮された映像を解凍して名前 $play$ により再生し，次の映像の待機に戻る．もし次の映像が 1 単位時間以内に配信されない場合は，名前 $attach$ により接続していないルータに接続要求を出し，r により接続しているルータと切断する．

システム全体は

$$\begin{aligned} System \stackrel{def}{=} {} & (\boldsymbol{\nu} attach, v, rel_1, rel_2) \\ & (Server \mid \overline{rel_1} \mid Router_1 \\ & \mid \overline{rel_2} \mid Router_2 \mid Player) \end{aligned}$$

と記述する．

システムの実行系列の一つを以下に示す.

$$
\begin{aligned}
&System\\
&\xrightarrow{\tau}{}^{6}\rightsquigarrow^{2}\cdots\\
&\xrightarrow{\tau}{}^{3}\rightsquigarrow\ (\boldsymbol{\nu} attach, v, rel_1, rel_2, g_1, g_2, g_3, u, l, h)\\
&\qquad (!\,\overline{v}\langle video\rangle \mid !\,(g_1.\cdots + g_1) \mid Router_1\\
&\qquad \mid attach(l).\cdots \mid Router_2 \mid !g_3(v,r).\cdots\\
&\qquad \mid !\,h.\cdots \mid t[\underline{0}].\overline{play}\langle video\rangle.\overline{g_3}\langle u, rel_1\rangle)\\
&\xrightarrow{\overline{play}}\xrightarrow{\tau}\ (\boldsymbol{\nu} attach, v, rel_1, rel_2, g_1, g_2, g_3, u, l, h)\\
&\qquad (!\,\overline{v}\langle video\rangle \mid !\,(g_1.\cdots + g_1) \mid Router_1\\
&\qquad \mid attach(l).\cdots \mid Router_2 \mid !\,h.\cdots\\
&\qquad \mid (u(video).\cdots.\overline{g_3}\langle u, rel_1\rangle + t[1].\cdots)\\
&\qquad \mid !\,g_3(v,r).\cdots)\\
&\quad\rightsquigarrow\quad (\boldsymbol{\nu} attach, v, rel_1, rel_2, g_1, g_2, g_3, u, l, h)\\
&\qquad (!\,\overline{v}\langle video\rangle \mid !\,(g_1.\cdots + g_1) \mid Router_1\\
&\qquad \mid attach(l).\cdots \mid Router_2 \mid !\,h.\cdots\\
&\qquad \mid \overline{attach}\langle l\rangle.\overline{rel_1}.\overline{h} \mid !\,g_3(v,r).\cdots)\\
&\xrightarrow{\tau}{}^{5}\rightsquigarrow^{4}\cdots\\
&\xrightarrow{\tau}{}^{5}\rightsquigarrow\ (\boldsymbol{\nu} attach, v, rel_1, rel_2, g_1, g_2, g_3, u, l, h)\\
&\qquad (!\,\overline{v}\langle video\rangle \mid \overline{u}\langle video\rangle.\overline{g_2} \mid !(g_2.\cdots + g_2)\\
&\qquad \mid !\,(g_1.\cdots + g_1) \mid Router_1 \mid Router_2\\
&\qquad \mid !\,h.\cdots \mid t[\underline{0}].\overline{play}\langle video\rangle.\overline{g_3}\langle u, rel_2\rangle\\
&\qquad \mid !\,g_3(v,r).\cdots)\\
&\quad\xrightarrow{\overline{play}}\quad \cdots
\end{aligned}
$$

この実行系列は以下の動作を表す.

（1） プレイヤが $Router_1$ に接続し映像を受信して再生

（2） 次の映像の待機中にタイムアウトが発生

（3） プレイヤは $Router_2$ に接続し次の映像を受信して再生

$\overline{play}$ によって表される映像の再生から次の再生までの遅延が最も短くなるのは，プレイヤが同じルータから続けて受信する場合で 1 単位時間の遅延が発生する．逆に遅延が最も長くなるのは，$Router_1$ に接続している状態で映像配信の待機中にタイムアウトし $Router_2$ に接続する場合で 6 単位時間の遅延が生じる．これらのことは

$$(\boldsymbol{\nu} v)(Server \mid !\,v(video).t[\underline{1}].\overline{play}\langle video\rangle) \lesssim_{\mathcal{T}} System$$

及び

$$System \lesssim_{\mathcal{T}} (\boldsymbol{\nu} v)(Server \mid !\,v(video).t[\underline{6}].\overline{play}\langle video\rangle)$$

を示すことで確認できる．$Server$ は時間独立であるため，定理 4 より

$$
\begin{aligned}
&!\,v(video).t[\underline{1}].\overline{play}\langle video\rangle\\
&\lesssim_{\mathcal{T}} (\boldsymbol{\nu} attach, \ldots)\\
&\quad (\overline{rel_1} \mid Router_1 \mid \overline{rel_2} \mid Router_2 \mid Player)
\end{aligned}
$$

及び

$$
\begin{aligned}
&(\boldsymbol{\nu} attach, \ldots)\\
&(\overline{rel_1} \mid Router_1 \mid \overline{rel_2} \mid Router_2 \mid Player)\\
&\lesssim_{\mathcal{T}} !\,v(video).t[\underline{6}].\overline{play}\langle video\rangle
\end{aligned}
$$

を示せばよい.

6. 関 連 研 究

Berger らはタイマ，メッセージ消失，プロセスの実行失敗などを導入して π 計算を拡張し，二相コミットプロトコルを記述している [5]．動作は時間ステップ関数と動作意味定義によって定義されており，時間ステップ関数の適用として時間の経過を表現する．時間経過を表す動作はなく，入出力動作や τ 動作による 1 ステップの遷移で 1 単位時間が経過する．Chen は π 計算に稠密時間を導入して拡張した [6]．更に選択演算子に対し保存される弱双模倣関係を定義し，弱合同性に対する完全な証明系を構築している．プロセスの振舞等価性は定義されているが，プロセスの相対的な速さの違いを反映する順序関係については考えられていない．これらの π 計算に対する時間拡張では我々の体系と異なり，時間待ちの長さをプロセス間通信によって直接送受信して動的に決定する仕組みはない.

CCS の体系においてはプロセスの “速さ” について研究がなされている．文献 [7], [8] では τ 遷移の回数を数え，回数が少ないほど速いプロセスであるとしている．CCS に対する時間拡張としては [9]〜[11] などがある．Timed CCS [9] では遅延動作，Temporal CCS [10] では時間制限付き選択演算子，TPL [11] ではタイムアウト演算子を導入して拡張している．Temporal CCS に対しては [12] で全体の時間経過量は同

じだが一方のプロセスが動作の実行タイミングが早いことを表す順序関係が定義されている．また文献 [13] では，すべての動作について初期状態からその動作を実行するまでに経過した時間が一方のプロセスが常に短い場合に成り立つ順序関係を定義している．本研究はこれらの CCS に基づいた体系の結果を π 計算に拡張し，リアルタイムオブジェクト指向言語の振舞いを記述するのに十分な動的な性質をもつ基本的な枠組みを提供した．CCS に基づく相対的な時間の意味論を本枠組みの上でも同様な形で展開できると考えられる．この点については今後の課題である．

7. むすび

本論文では，文献 [1] に基づく π 計算の時間拡張において，双模倣関係が入力プレフィックスを除くコンテクストに対して合同であることを示した．双模倣関係は同じ入力や出力などの動作が同じタイミングで実行されることを表す．更に入出力動作が実行されるタイミングの違いを表すために遅延時間順関係を定義した．この体系により [1] で示すようにリアルタイムオブジェクト指向言語に対し意味記述を与えることができる．特に時間待ちの長さを通信できる点が有用である．

実時間システムの実装が仕様よりも速く動作することが示されるならば，実装がデッドラインを満たしながら動作することが従来の手法 [14] と同様にして示される．我々の時間拡張において，タイムアウト動作は時間経過動作と選択演算子によりモデル化される．遅延時間順関係がチャネル名を束縛する入力プレフィックス以外のコンテクストに対して合同であるため，タイムアウト動作を付け加えても関係は保存される．この点において弱遅延時間順関係は実時間並行プログラムの正しさを示す手法の基礎として有用である．弱遅延時間順関係は時間経過によって変化しないプロセスの合成に対しては保存される．この制約はサーバクライアントシステムに対しては多くの場合大きな問題とはならない．なぜならストリーミングシステムの例で示されたように，サーバは通常クライアントの要求を処理するプロセスであり，その意味で時間経過による変化はしないからである．

本論文では，時間の上限についての関係を示した．しかし，実時間システムの動作の正しさを保証するためには，時間の下限も重要な意味をもつ．例えば，ある動作を 5 秒以上 15 秒以内に行うプロセスは，同じ動作を 7 秒以上 12 秒以内に行うプロセスを包含しているといえる．このような問題に対するテスト等価性 [15] などの手法を用いた検証手法の開発については今後の課題である．

遅延時間順関係はある実行系列において任意の二つの入出力動作の間の経過時間が一方のプロセスの方が必ず短いことを示す関係であるが，実際の実時間システムでは特定の入出力動作の間の経過時間や実行開始からの経過時間について大小関係を示せば十分な場合も多い．また時間経過により状態が変化するプロセスの並行合成に対し保存されないため，時間制約をもつコンポーネントの組合せに適用できない．文献 [13], [16] で提案される初期状態からの経過時間によって順序づけるような関係について検討することは今後の課題である．

謝辞 熱心に議論して頂いた阿草研究室の皆様に感謝致します．本研究の一部は，科学研究費補助金基盤研究 (B) 17300006，基盤研究 (C) 16500027，及び，栢森情報科学振興財団の助成による．

文献

[1] 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，” 情処学論，vol.45, no.6, pp.1498–1507, 2004.

[2] R. Milner, Communication and Mobile Systems: The π-Calculus, Cambridge University Press, 1999.

[3] R. Milner, J. Parrow, and D. Walker, “A calculus of mobile processes, part I/II,” Inf. Comput., vol.100, no.1, pp.1–77, 1992.

[4] D. Sangiorgi and D. Walker, The π-calculus: A Theory of Mobile Processes, Cambridge University Press, 2001.

[5] M. Berger and K. Honda, “The two-phase commitment protocol in an extended π-calculus,” Electronic Notes in Theoretical Computer Science, vol.39, no.1, pp.21–46, Elsevier, 2003.

[6] J. Chen, “A proof system for weak congruence in timed π calculus,” Technical Report, Laboratoire d’Informatique Fondamentale d’Orléans, Université d’Orléans, pp.1–28, France, 2004.

[7] S. Arun-Kumar and M. Hennessy, “An efficiency preorder for processes,” Acta Informatica, vol.29, no.8, pp.737–760, 1992.

[8] V. Natarajan and R. Cleaveland, “An algebraic theory of process efficiency,” LICS ’96, pp.63–72, IEEE Computer Society Press, 1996.

[9] W. Yi, A Calculus of Real Time Systems, PhD Thesis, Chalmers University of Technology, 1991.

[10] F. Moller and C. Tofts, “A temporal calculus of communicating systems,” CONCUR ’90, vol.458 of LNCS, pp.401–415, Springer, 1990.

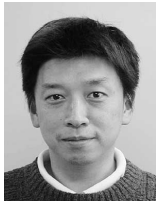
[11] M. Hennessy and T. Regan, “A process algebra for

timed systems," Inf. Comput., vol.117, no.2, pp.221–239, 1995.
[12] F. Moller and C. Tofts, "Relating processes with respect to speed," CONCUR '91, vol.527 of LNCS, pp.424–438, Springer, 1991.
[13] G. Lüttgen and W. Vogler, "Bisimulation on speed: Worst-case efficiency," Inf. Comput., vol.191, no.2, pp.105–144, 2004.
[14] R. Milner, Communication and Concurrency, Cambridge University Press, 1988.
[15] M. Boreale and R.D. Nicola, "Testing equivalence for mobile processes," Inf. Comput., vol.120, no.2, pp.279–303, 1995.
[16] G. Lüttgen and W. Vogler, "Bisimulations on speed: Lower time bounds," Theoretical Informatics and Applications, vol.39, pp.587–618, 2005.

（平成 17 年 7 月 19 日受付，11 月 4 日再受付）

桑原　寛明

2001 名大・工・電気電子情報卒．2003 同大大学院工学研究科計算理工学専攻博士前期課程了．現在，同大学院情報科学研究科情報システム学専攻博士後期課程在学中．プロセス代数，形式的手法を用いた実時間システム開発に関する研究に興味をもつ．情報処理学会，日本ソフトウェア科学会各学生員．



結縁　祥治　（正員）

平 2 名古屋大大学院博士課程了．名古屋大学工学研究科助手，平 10 同情報メディア教育センター助教授を経て，現在，同大学院情報科学研究科助教授．博士（工学）．並行計算モデルのソフトウェアへの応用面の観点で，通信プロセスモデルの研究に従事．形式的な手法によるモデル化に基づくソフトウェア検証に興味をもつ．

阿草　清滋　（正員）

1970 京大・工・電気第二学科卒．1972 同大大学院工学研究科電気工学第二専攻修士課程了．同博士課程へ進学．1974 より同情報工学科助手．同講師，助教授を経て 1989 より名古屋大学教授．現在，同大学院情報科学研究科教授．工博．専門分野はソフトウェア工学，ソフトウェア開発方法論，知的開発環境，ソフトウェアデータベース，仕様化技法，再利用技法，マンマシンインタフェース．情報処理学会，ソフトウェア科学会，IEEE，ACM 各会員．